

Geodetic Line at Constant Altitude above the Ellipsoid

Richard J. Mathar*

Leiden Observatory, Leiden University, P.O. Box 9513, 2300 RA Leiden, The Netherlands[†]

(Dated: May 2, 2008)

The two-dimensional surface of a bi-axial ellipsoid is characterized by the lengths of its major and minor axes. Longitude and latitude span an angular coordinate system across. We consider the egg-shaped surface of constant altitude above (or below) the ellipsoid surface, and compute the geodetic lines—lines of minimum Euclidean length—within this surface which connect two points of fixed coordinates. This addresses the common “inverse” problem of geodesics generalized to non-zero elevations. The system of differential equations which couples the two angular coordinates along the trajectory is reduced to a single integral, which is handled by Taylor expansion up to fourth power in the eccentricity.

PACS numbers: 91.10.By, 91.10.Ws, 02.40.Hw

Keywords: geodesy; ellipsoid; eccentricity; geodetic line; inverse problem

I. CONTENTS

The common three parameters employed to relate Cartesian coordinates to an ellipsoidal surface are the angles of latitude and longitude in a grid on the surface, plus an altitude which is a shortest (perpendicular) distance to the surface. The well-known functional relations (coordinate transformations) are summarized in Section II.

The inverse problem of geodesy is to find the line embedded in the ellipsoid surface which connects two fixed points subject to minimization of its length. We pose the equivalent problem for lines at constant altitude, as if one would ask for the shortest track of the center of a sphere of given radius which rolls on the ellipsoid surface and meets two points of the same, known altitude. In Section III, we formulate this in terms of the generic differential equations of geodesy parametrized by the Christoffel symbols.

In the main Section IV, the coupled system of differential equations of latitude and longitude as a function of path length is reduced to one degree of freedom, here chosen to be the direction along the path at one of the fixed terminal points, measured in the topocentric horizontal system, and dubbed the launching angle. Closed-form expressions of this parameter in terms of the coordinates of the terminal points have not been found; instead, the results are presented as series expansions up to fourth power in the eccentricity.

The standard treatment of this analysis is the projection on an auxiliary sphere; this technique is (almost) completely ignored but for the pragmatic aspect that the case of zero eccentricity is a suitable zeroth-order reference of series expansions around small eccentricities.

II. SPHEROIDAL COORDINATES

A. Surface

The cross section of an ellipsis of equatorial radius $\rho_e > \rho_p$ with eccentricity e in a Cartesian (x, z) system is:

$$\rho_p^2 = \rho_e^2(1 - e^2); \quad \frac{x^2}{\rho_e^2} + \frac{z^2}{\rho_p^2} = 1. \quad (1)$$

The ellipsis defines a geocentric latitude ϕ' and a geodetic latitude ϕ , the latter measured by intersection of the normal to the tangential plane with the equatorial plane (Figure 1). On the surface of the ellipsoid [19, §IX][6][2, §140]:

$$\tan \phi = \frac{z}{x} \frac{\rho_e^2}{\rho_p^2}; \quad \frac{z}{x} = \tan \phi' = \frac{\rho_p^2}{\rho_e^2} \tan \phi. \quad (2)$$

*Electronic address: mathar@strw.leidenuniv.nl; URL: <http://www.strw.leidenuniv.nl/~mathar>

[†]This work is supported by the NWO VICI grant 639.043.201 “Optical Interferometry: A new Method for Studies of Extrasolar Planets” to A. Quirrenbach.

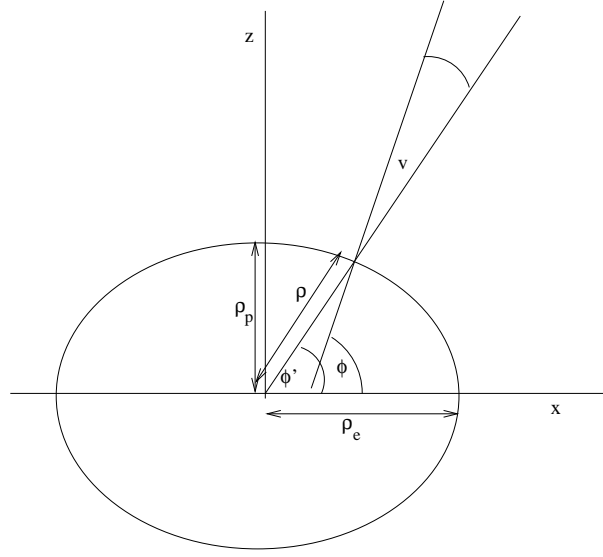


FIG. 1: Major semi-axis ρ_e , minor semi-axis ρ_p , straight distance to the center of coordinates ρ , geocentric latitude ϕ' , geodetic latitude ϕ , and their pointing difference v .

Supposed ρ denotes the distance from the center of coordinates to the point on the geoid surface, the transformation from (x, z) to ϕ is

$$x = \rho \cos \phi' = \frac{\rho_e \cos \phi}{\sqrt{1 - e^2 \sin^2 \phi}}, \quad z = \rho \sin \phi' = \frac{\rho_e (1 - e^2) \sin \phi}{\sqrt{1 - e^2 \sin^2 \phi}}. \quad (3)$$

$v = \phi - \phi'$ defines the difference between the geodetic (astronomical) and the geocentric latitudes,

$$\tan v = \frac{e^2 \sin(2\phi)}{2(1 - e^2 \sin^2 \phi)} = \frac{m \sin(2\phi)}{1 + m \cos(2\phi)} = m \sin(2\phi) - m^2 \sin(2\phi) \cos(2\phi) + m^3 \sin(2\phi) \cos^2(2\phi) + \dots; \quad (4)$$

where the expansion of the denominator has been given in terms of the geometric series of the parameter

$$m \equiv \frac{e^2}{2 - e^2}. \quad (5)$$

The Taylor series in powers of $\sin(2\phi)$ and in powers of m are

$$v = \frac{m}{1 + m} \sin(2\phi) + \frac{m^2(2 + m)}{6(1 + m)^3} \sin^3(2\phi) + \frac{m^2(5 + 25m + 15m^2 + 3m^3)}{40(1 + m)^5} \sin^5(2\phi) + \dots \quad (6)$$

$$= \sin(2\phi)m - \frac{1}{2} \sin(4\phi)m^2 + \frac{1}{3} \sin(6\phi)m^3 - \frac{1}{4} \sin(8\phi)m^4 + \frac{1}{5} \sin(10\phi)m^5 + \dots. \quad (7)$$

A flattening factor f is also commonly defined [11],

$$f = \frac{\rho_e - \rho_p}{\rho_e} = 1 - \sqrt{1 - e^2}; \quad e^2 = f(2 - f). \quad (8)$$

The reference values of the Earth ellipsoid adopted in the WGS84 [14] are

$$f = 1/298.257223563, \quad \rho_e = 6378137.0 \text{ m}. \quad (9)$$

The numerical evaluation of (6) in terms of a Fourier series with this parametrization is, in units of radians and arcseconds:

$$v = 0.0033584338 \sin(2\phi) - 0.56395388 \times 10^{-5} \sin(4\phi) + 0.12626678 \times 10^{-7} \sin(6\phi) + \dots \quad (10)$$

$$= 692.''72669 \sin(2\phi) - 1.''16324 \sin(4\phi) + 0.''00260 \sin(6\phi) + \dots. \quad (11)$$

Slightly different values would emerge according to IERS conventions of 2003 [13].

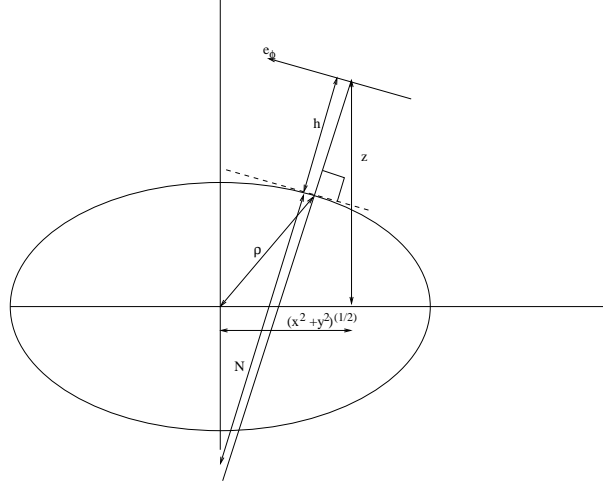


FIG. 2: A point with Cartesian coordinates (x, y, z) has a distance z to the equatorial plane, a distance $\sqrt{x^2 + y^2}$ to the polar axis, a distance h to the surface of the ellipsoid, and a distance $N + h$ to the polar axis, measured along the local normal to the surface [12]. The vector \mathbf{e}_ϕ points North at this point.

B. General Altitude

If one moves along the direction of ϕ a distance h away from the surface of the geoid, the new coordinates relative to (3) are [7, 9, 10, 16, 23, 25]

$$x = \rho \cos \phi' + h \cos \phi; \quad z = \rho \sin \phi' + h \sin \phi, \quad (12)$$

which can be written in terms of a distance $N(\phi)$,

$$N(\phi) \equiv \frac{\rho_e}{\sqrt{1 - e^2 \sin^2 \phi}} \quad (13)$$

as

$$x = (N + h) \cos \phi; \quad z = [N(1 - e^2) + h] \sin \phi. \quad (14)$$

Rotation of (14) around the polar axis with geographic longitude λ defines the full 3D transformation between (x, y, z) and (λ, ϕ, h) ,

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} [N(\phi) + h] \cos \phi \cos \lambda \\ [N(\phi) + h] \cos \phi \sin \lambda \\ [N(\phi)(1 - e^2) + h] \sin \phi \end{pmatrix}. \quad (15)$$

The only theme of this paper is to generalize the geodetic lines of the literature [3, 6, 15, 18, 20, 21] to the case of finite altitude $h \neq 0$. The physics of gravimetric or potential theory is not involved, only the mathematics of the geometry. It should be noted that points with constant, non-zero h do *not* define a surface of an ellipsoid with effective semi-axes $\rho_{e,p} + h$ —otherwise the geodetic line could be deduced by mapping the problem onto an equivalent ellipsoidal surface [6].

III. INVERSE PROBLEM OF GEODESY

A. Topocentric Coordinate System

A line of shortest distance at constant height $h = \text{const}$ between two points 1 and 2 is defined by minimizing the Euclidean distance, the line integral

$$S = \int_1^2 \sqrt{dx^2 + dy^2 + dz^2} = \int_1^2 \sqrt{ds^2} \equiv \int_1^2 \mathcal{L} ds \quad (16)$$

for some parametrization $\lambda(\phi)$. The integrand is equivalent to a Lagrange function $\mathcal{L}(\phi, \lambda, d\lambda/d\phi)$ or $\mathcal{L}(\phi, \lambda, d\phi/d\lambda)$.

The gradient of (15) with respect to λ and ϕ defines the vectors $\mathbf{e}_{\lambda, \phi}$ that span the topocentric tangent plane

$$\mathbf{e}_\lambda = \begin{pmatrix} -[N(\phi) + h] \cos \phi \sin \lambda \\ [N(\phi) + h] \cos \phi \cos \lambda \\ 0 \end{pmatrix}; \quad \mathbf{e}_\phi = \begin{pmatrix} -\sin \phi \cos \lambda [M(\phi) + h] \\ -\sin \phi \sin \lambda [M(\phi) + h] \\ \cos \phi [M(\phi) + h] \end{pmatrix}, \quad (17)$$

where a meridional radius of curvature [22]

$$M(\phi) \equiv \frac{\rho_e(1 - e^2)}{(1 - e^2 \sin^2 \phi)^{3/2}} = N(\phi) \frac{1 - e^2}{1 - e^2 \sin^2 \phi} \quad (18)$$

is defined to simplify the notation. Building squares and dot products computes the three Gauss Fundamental parameters of the surface [17]

$$\mathbf{e}_\lambda^2 = E = (N + h)^2 \cos^2 \phi; \quad (19)$$

$$\mathbf{e}_\lambda \cdot \mathbf{e}_\phi = F = 0; \quad (20)$$

$$\mathbf{e}_\phi^2 = G = \left[N \frac{1 - e^2}{1 - e^2 \sin^2 \phi} + h \right]^2 = (M + h)^2. \quad (21)$$

Specializing to $h = 0$ we get the You formulae [24]. E and G provide the principal curvatures along the meridian and azimuth [5], and the coefficients of the metric tensor in the quadratic form of ds^2 ,

$$S = \int \sqrt{Ed\lambda^2 + 2Fd\lambda d\phi + Gd\phi^2}. \quad (22)$$

B. Christoffel Symbols

Christoffel symbols are the connection coefficients between differentials $d\mathbf{e}_\epsilon$ of the topocentric axis and of the positions $d\mathbf{x}^\beta$, in a generic definition

$$d\mathbf{e}_\epsilon = \sum_{\alpha, \beta} \mathbf{e}_\alpha \Gamma_{\beta\epsilon}^\alpha d\mathbf{x}^\beta. \quad (23)$$

This format is matched by first computing the derivative of \mathbf{e}_λ with respect to λ and ϕ (at $h = \text{const}$):

$$d\mathbf{e}_\lambda = \begin{pmatrix} -(N + h) \cos \phi \cos \lambda \\ -(N + h) \cos \phi \sin \lambda \\ 0 \end{pmatrix} d\lambda + \begin{pmatrix} (M + h) \sin \phi \sin \lambda \\ -(M + h) \sin \phi \cos \lambda \\ 0 \end{pmatrix} d\phi \quad (24)$$

and of \mathbf{e}_ϕ with respect to λ and ϕ :

$$d\mathbf{e}_\phi = \begin{pmatrix} (M + h) \sin \phi \sin \lambda \\ -(M + h) \sin \phi \cos \lambda \\ 0 \end{pmatrix} d\lambda + \begin{pmatrix} \left[Ne^2 \frac{\cos^2 \phi}{1 - e^2 \sin^2 \phi} - 3Ne^2(1 - e^2) \frac{\sin^2 \phi}{(1 - e^2 \sin^2 \phi)^2} - (N + h) \right] \cos \phi \cos \lambda \\ \left[Ne^2 \frac{\cos^2 \phi}{1 - e^2 \sin^2 \phi} - 3Ne^2(1 - e^2) \frac{\sin^2 \phi}{(1 - e^2 \sin^2 \phi)^2} - (N + h) \right] \cos \phi \sin \lambda \\ \left[Ne^2(1 - e^2) \left(3 \frac{\cos^2 \phi}{(1 - e^2 \sin^2 \phi)^2} - \frac{\sin^2 \phi}{1 - e^2 \sin^2 \phi} \right) - [N(1 - e^2) + h] \right] \sin \phi \end{pmatrix} d\phi. \quad (25)$$

The next step splits these two equations to the expanded version of (23),

$$d\mathbf{e}_\lambda = (\mathbf{e}_\lambda \Gamma_{\lambda\lambda}^\lambda + \mathbf{e}_\phi \Gamma_{\lambda\lambda}^\phi) d\lambda + (\mathbf{e}_\lambda \Gamma_{\phi\lambda}^\lambda + \mathbf{e}_\phi \Gamma_{\phi\lambda}^\phi) d\phi; \quad (26)$$

$$d\mathbf{e}_\phi = (\mathbf{e}_\lambda \Gamma_{\lambda\phi}^\lambda + \mathbf{e}_\phi \Gamma_{\lambda\phi}^\phi) d\lambda + (\mathbf{e}_\lambda \Gamma_{\phi\phi}^\lambda + \mathbf{e}_\phi \Gamma_{\phi\phi}^\phi) d\phi. \quad (27)$$

The eight Γ are extracted by evaluating dot products of the four vector coefficients in (24)–(25) by \mathbf{e}_λ and \mathbf{e}_ϕ ,

$$\Gamma_{\lambda\lambda}^\lambda = \Gamma_{\phi\lambda}^\phi = \Gamma_{\lambda\phi}^\phi = \Gamma_{\phi\phi}^\phi = 0; \quad (28)$$

$$\Gamma_{\lambda\lambda}^\phi = (N + h) \cos \phi \sin \phi \frac{1 - e^2 \sin^2 \phi}{h(1 - e^2 \sin^2 \phi) + N(1 - e^2)}; \quad (29)$$

$$\Gamma_{\phi\lambda}^\lambda = \Gamma_{\lambda\phi}^\lambda = -\sin \phi \frac{h(1 - e^2 \sin^2 \phi) + N(1 - e^2)}{(N + h)(1 - e^2 \sin^2 \phi) \cos \phi}; \quad (30)$$

$$\Gamma_{\phi\phi}^\phi = 3Ne^2(1 - e^2) \sin \phi \cos \phi \frac{1}{[h(1 - e^2 \sin^2 \phi) + N(1 - e^2)](1 - e^2 \sin^2 \phi)}. \quad (31)$$

The Euler-Lagrange Differential Equations $\delta \int_1^2 \sqrt{Ed\lambda^2 + Gd\phi^2} = 0$ for a stationary Lagrange density \mathcal{L} (at $F = 0$) become the differential equations of the geodesic [17], in the generic format

$$\frac{d^2 x^\epsilon}{ds^2} + \sum_{\mu\nu} \Gamma_{\mu\nu}^\epsilon \frac{dx^\mu}{ds} \frac{dx^\nu}{ds} = 0. \quad (32)$$

The explicit write-up

$$\frac{d^2 \lambda}{ds^2} + \Gamma_{\lambda\lambda}^\lambda \frac{d\lambda}{ds} \frac{d\lambda}{ds} + \Gamma_{\lambda\phi}^\lambda \frac{d\lambda}{ds} \frac{d\phi}{ds} + \Gamma_{\phi\lambda}^\lambda \frac{d\phi}{ds} \frac{d\lambda}{ds} + \Gamma_{\phi\phi}^\lambda \frac{d\phi}{ds} \frac{d\phi}{ds} = 0; \quad (33)$$

$$\frac{d^2 \phi}{ds^2} + \Gamma_{\lambda\lambda}^\phi \frac{d\lambda}{ds} \frac{d\lambda}{ds} + \Gamma_{\lambda\phi}^\phi \frac{d\lambda}{ds} \frac{d\phi}{ds} + \Gamma_{\phi\lambda}^\phi \frac{d\phi}{ds} \frac{d\lambda}{ds} + \Gamma_{\phi\phi}^\phi \frac{d\phi}{ds} \frac{d\phi}{ds} = 0, \quad (34)$$

simplifies with (28) to

$$\frac{d^2 \lambda}{ds^2} + 2\Gamma_{\lambda\phi}^\lambda \frac{d\lambda}{ds} \frac{d\phi}{ds} = 0; \quad (35)$$

$$\frac{d^2 \phi}{ds^2} + \Gamma_{\lambda\lambda}^\phi \left(\frac{d\lambda}{ds} \right)^2 + \Gamma_{\phi\phi}^\phi \left(\frac{d\phi}{ds} \right)^2 = 0. \quad (36)$$

IV. REDUCTION OF THE DIFFERENTIAL EQUATIONS

A. Separation of Angular Variables

Decoupling of the two differential equations (35)–(36) starts with the separation of variables in (35),

$$\frac{\frac{d^2 \lambda}{ds^2}}{\frac{d\lambda}{ds}} = -2\Gamma_{\phi\lambda}^\lambda \frac{d\phi}{ds}. \quad (37)$$

Change of the integration variable on the right hand side from s to ϕ allows to use the underivative of (30)

$$\int \Gamma_{\lambda\phi}^\phi(\phi) d\phi = \log \{ [N(\phi) + h] \cos \phi \} + \text{const} \quad (38)$$

to generate a first integral

$$\log \frac{d\lambda}{ds} = -2 \log [(N + h) \cos \phi] + \text{const}. \quad (39)$$

Exponentiation yields

$$\frac{d\lambda}{ds} = c_3 \frac{1}{(N + h)^2 \cos^2 \phi}, \quad (40)$$

where

$$c_3 \equiv \frac{d\lambda}{ds} \Big|_1 (N_1 + h)^2 \cos^2 \phi_1 \quad (41)$$

is a constant for each geodesic. It has been defined with the azimuth ϕ_1 and $N_1 \equiv N(\phi_1)$ at the start of the line, but could as well be associated with any other point or the end point 2. c_3 is positive for trajectories starting into eastwards direction, negative for the westwards heading, zero for routes to the poles. Moving the square of (40) into (36) yields

$$\frac{d^2\phi}{ds^2} + \frac{\sin\phi}{(N+h)^3 \cos^3\phi} \frac{1-e^2 \sin^2\phi}{h(1-e^2 \sin^2\phi) + N(1-e^2)} c_3^2 + \frac{3Ne^2(1-e^2) \sin\phi \cos\phi}{[h(1-e^2 \sin^2\phi) + N(1-e^2)](1-e^2 \sin^2\phi)} \left(\frac{d\phi}{ds}\right)^2 = 0. \quad (42)$$

To solve this differential equation, we substitute the variable ϕ by its projection τ onto the polar axis,

$$\tau \equiv \sin\phi, \quad (43)$$

which implies transformations in the derivatives:

$$\frac{d\tau}{ds} = \cos\phi \frac{d\phi}{ds}; \quad (44)$$

$$\frac{d^2\tau}{ds^2} = -\sin\phi \frac{d\phi}{ds} \frac{d\phi}{ds} + \cos\phi \frac{d^2\phi}{ds^2}; \quad (45)$$

$$\cos\phi \frac{d^2\phi}{ds^2} = \frac{d^2\tau}{ds^2} + \tau \left(\frac{d\phi}{ds}\right)^2 = \frac{d^2\tau}{ds^2} + \frac{\tau}{\cos^2\phi} \left(\frac{d\tau}{ds}\right)^2 = \frac{d^2\tau}{ds^2} + \frac{\tau}{1-\tau^2} \left(\frac{d\tau}{ds}\right)^2. \quad (46)$$

We multiply (42) by $\cos\phi$, then replace $d\phi/ds$ and $d^2\phi/ds^2$ as noted above,

$$\frac{d^2\tau}{ds^2} + \frac{\tau}{1-\tau^2} \left(\frac{d\tau}{ds}\right)^2 + \frac{\tau}{(N+h)^3(1-\tau^2)} \frac{1-e^2\tau^2}{h(1-e^2\tau^2) + N(1-e^2)} c_3^2 + \frac{3Ne^2(1-e^2)\tau}{[h(1-e^2\tau^2) + N(1-e^2)](1-e^2\tau^2)} \left(\frac{d\tau}{ds}\right)^2 = 0;$$

$$\begin{aligned} \frac{d^2\tau}{ds^2} + \frac{\tau}{1-\tau^2} \frac{h(1-e^2\tau^2)^2 + N(1-e^2)(1+3e^2-4e^2\tau^2)}{[h(1-e^2\tau^2) + N(1-e^2)][1-e^2\tau^2]} \left(\frac{d\tau}{ds}\right)^2 \\ + \frac{\tau}{(N+h)^3(1-\tau^2)} \frac{1-e^2\tau^2}{h(1-e^2\tau^2) + N(1-e^2)} c_3^2 = 0. \end{aligned} \quad (47)$$

This is a differential equation with no explicit appearance of the independent variable s ,

$$(1-\tau^2) \frac{d^2\tau}{ds^2} + \frac{h(1-e^2\tau^2)^2 + N(1-e^2)(1+3e^2-4e^2\tau^2)}{[h(1-e^2\tau^2) + N(1-e^2)][1-e^2\tau^2]} \tau \left(\frac{d\tau}{ds}\right)^2 + \frac{\tau c_3^2}{(N+h)^3} \frac{1-e^2\tau^2}{h(1-e^2\tau^2) + N(1-e^2)} = 0,$$

and the standard way of progressing is the substitution

$$\frac{d\tau}{ds} \equiv p; \quad \frac{d^2\tau}{ds^2} = p \frac{dp}{d\tau}; \quad (48)$$

$$(1-\tau^2) p \frac{dp}{d\tau} + \frac{h(1-e^2\tau^2)^2 + N(1-e^2)(1+3e^2-4e^2\tau^2)}{[h(1-e^2\tau^2) + N(1-e^2)][1-e^2\tau^2]} \tau p^2 + \frac{\tau c_3^2}{(N+h)^3} \frac{1-e^2\tau^2}{h(1-e^2\tau^2) + N(1-e^2)} = 0. \quad (49)$$

This is transformed to a linear differential equation by the further substitution $P \equiv p^2$, $dP/d\tau = 2p dp/d\tau$,

$$\frac{1}{2}(1-\tau^2) \frac{dP}{d\tau} + \frac{h(1-e^2\tau^2)^2 + N(1-e^2)(1+3e^2-4e^2\tau^2)}{[h(1-e^2\tau^2) + N(1-e^2)][1-e^2\tau^2]} \tau P + \frac{\tau c_3^2}{(N+h)^3} \frac{1-e^2\tau^2}{h(1-e^2\tau^2) + N(1-e^2)} = 0. \quad (50)$$

The standard approach is to solve the homogeneous differential equation first,

$$\frac{dP}{d\tau} = -2\tau \frac{h(1-e^2\tau^2)^2 + N(1-e^2)(1+3e^2-4e^2\tau^2)}{[1-\tau^2][h(1-e^2\tau^2) + N(1-e^2)][1-e^2\tau^2]} P. \quad (51)$$

After division through P , the left hand side is easily integrated, and the right hand side (incompletely) decomposed into partial fractions,

$$\begin{aligned}\log P &= -2 \int \tau \frac{h(1 - e^2 \tau^2)^2 + N(1 - e^2)(1 + 3e^2 - 4e^2 \tau^2)}{[1 - \tau^2][h(1 - e^2 \tau^2) + N(1 - e^2)][1 - e^2 \tau^2]} d\tau \\ &= \int \frac{-2\tau}{1 - \tau^2} d\tau + 3 \int \frac{-2e^2 \tau}{1 - e^2 \tau^2} d\tau + 6he^2 \int \frac{\tau}{h(1 - e^2 \tau^2) + N(1 - e^2)} d\tau \\ &= \log(1 - \tau^2) + 3 \log(1 - e^2 \tau^2) - 2 \log \left[h(1 - e^2 \tau^2)^{3/2} + \rho_e(1 - e^2) \right] + \text{const.}\end{aligned}\quad (52)$$

$$P = \text{const.} \cdot \frac{(1 - \tau^2)(1 - e^2 \tau^2)^3}{[h(1 - e^2 \tau^2)^{3/2} + \rho_e(1 - e^2)]^2} = \text{const.} \cdot \frac{(1 - \tau^2)(1 - e^2 \tau^2)^2}{[h(1 - e^2 \tau^2) + N(1 - e^2)]^2} = \frac{\text{const.} \cdot (1 - \tau^2)}{G(\tau)}.$$

Solution of the *inhomogeneous* differential equation (50) proceeds with the variation of the constant, the ansatz

$$P = c(\tau) \cdot \frac{(1 - \tau^2)(1 - e^2 \tau^2)^2}{[h(1 - e^2 \tau^2) + N(1 - e^2)]^2}. \quad (53)$$

Back insertion into (50) leads to a first order differential equation for $c(\tau)$,

$$\frac{dc(\tau)}{d\tau} \frac{1 - \tau^2}{G(\tau)} = - \frac{2\tau c_3^2}{(1 - \tau^2)(N + h)^3} \frac{1 - e^2 \tau^2}{h(1 - e^2 \tau^2) + N(1 - e^2)},$$

which is decomposed into partial fractions

$$\frac{dc(\tau)}{d\tau} = c_3^2 \left[eN \frac{2e\tau}{1 - e^2 \tau^2} \frac{1}{(N + h)^3} + \frac{-2\tau}{1 - \tau^2} \frac{1}{(N + h)^2} \right] \frac{1}{1 - \tau^2}.$$

The ensuing integral over $d\tau$ is solved by aid of the substitution $\tau^2 = u$,

$$c(\tau) = -c_3^2 \frac{1}{(N + h)^2(1 - \tau^2)} + \text{const.}$$

Back into (53)—using *const* to indicate placement of any member of an anonymous bag of constants of integration,

$$P = \left[\text{const} - \frac{1}{(N + h)^2(1 - \tau^2)} \right] \frac{c_3^2(1 - \tau^2)(1 - e^2 \tau^2)^2}{[h(1 - e^2 \tau^2) + N(1 - e^2)]^2} \quad (54)$$

$$= c_5 \frac{(1 - \tau^2)(1 - e^2 \tau^2)^2}{[h(1 - e^2 \tau^2) + N(1 - e^2)]^2} - \frac{c_3^2(1 - e^2 \tau^2)^2}{(N + h)^2 [h(1 - e^2 \tau^2) + N(1 - e^2)]^2} \quad (55)$$

$$= c_5 \frac{1}{(h + M)^2} \left(1 - \tau^2 - \frac{c_3^2}{(N + h)^2} \right) = c_5 \frac{1 - \tau^2}{G(\tau)} \left(1 - \frac{c_3^2}{E(\tau)} \right) = p^2. \quad (56)$$

The subscript 1 denotes values at the starting point of the curve,

$$P_1 = c_5 \frac{\cos^2 \phi_1 (1 - e^2 \sin^2 \phi_1)^2}{[h(1 - e^2 \sin^2 \phi_1) + N_1(1 - e^2)]^2} - \frac{c_3^2(1 - e^2 \sin^2 \phi_1)^2}{(N_1 + h)^2 [h(1 - e^2 \sin^2 \phi_1) + N_1(1 - e^2)]^2} = p_1^2 \quad (57)$$

$$= c_5 \frac{\cos^2 \phi_1 (1 - e^2 \sin^2 \phi_1)^2}{[h(1 - e^2 \sin^2 \phi_1) + N_1(1 - e^2)]^2} - \frac{(d\lambda/ds)_1^2 (N_1 + h)^2 \cos^4 \phi_1 (1 - e^2 \sin^2 \phi_1)^2}{[h(1 - e^2 \sin^2 \phi_1) + N_1(1 - e^2)]^2}. \quad (58)$$

Solving for c_5 yields

$$\begin{aligned}c_5 &= \left(\frac{d\lambda}{ds|_1} \right)^2 \cos^2 \phi_1 (N_1 + h)^2 + \frac{p_1^2 [h(1 - e^2 \sin^2 \phi_1) + N_1(1 - e^2)]^2}{\cos^2 \phi_1 (1 - e^2 \sin^2 \phi_1)^2} = \left(\frac{d\lambda}{ds|_1} \right)^2 \cos^2 \phi_1 (N_1 + h)^2 + \frac{p_1^2}{\cos^2 \phi_1} G_1 \\ &= \left(\frac{d\lambda}{ds|_1} \right)^2 E_1 + \left(\frac{d\phi}{ds|_1} \right)^2 G_1 = \frac{c_3^2}{(N_1 + h)^2 \cos^2 \phi_1} + \left(\frac{d\phi}{ds|_1} \right)^2 (M_1 + h)^2.\end{aligned}\quad (59)$$

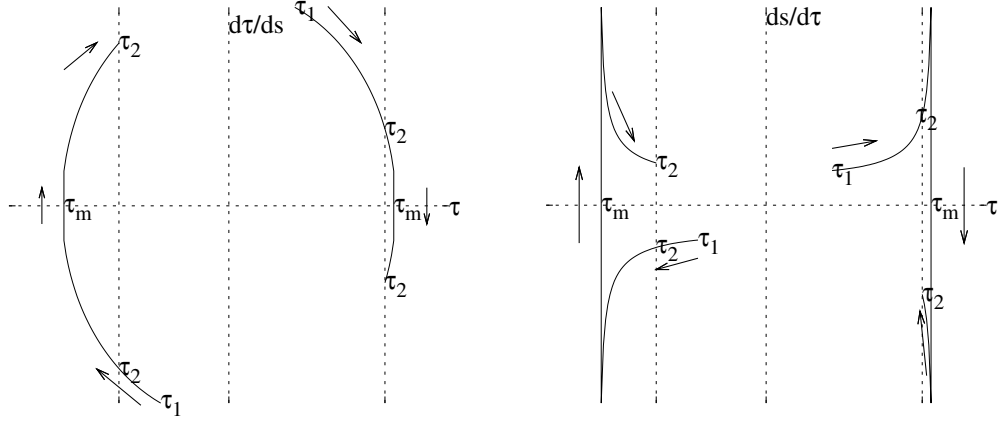


FIG. 3: Two examples of trajectories of $d\tau/ds$ [left, equation (61)] or $ds/d\tau$ [right] as a function of τ . They may or may not pass through a zero τ_m of (61) while connecting the starting abscissa τ_1 with the final abscissa τ_2 . There are four basic topologies, depending on whether τ_m is positive or negative, and depending on whether the sign change in $d\tau/ds$ is from $+$ to $-$ or from $-$ to $+$ at this point.

Compared with the differential version of (22),

$$ds^2 = E d\lambda^2 + G d\phi^2; \quad 1 = E \left(\frac{d\lambda}{ds} \right)^2 + G \left(\frac{d\phi}{ds} \right)^2, \quad (60)$$

we must have $c_5 = 1$. Note that (56) is essentially a write-up for $p^2 \sim (d\phi/ds)^2$ and could be derived quickly by inserting (40) directly into (60).

The square root of (56) is

$$p = \pm \frac{1 - e^2 \tau^2}{h(1 - e^2 \tau^2) + N(1 - e^2)} \sqrt{1 - \tau^2 - \frac{c_3^2}{(N+h)^2}} = \pm \frac{\sqrt{1 - \tau^2 - \frac{c_3^2}{(N+h)^2}}}{h + M(\tau)} = \frac{d\tau}{ds}. \quad (61)$$

The sign in front of the square root is to be chosen positive for pieces of the trajectory with $d\tau/ds > 0$ (northern direction), negative where $d\tau/ds < 0$ (southern direction). The value in the square root may run through zero within one curve, $d\tau/ds = 0$ at one point τ_m , such that the square root switches sign there (Figure 3). This happens whenever

$$(1 - \tau_m^2)[N(\tau_m) + h]^2 = c_3^2 \quad (62)$$

yields a vanishing discriminant of the square root for some (λ, ϕ) , that is, whenever the difference $\lambda_2 - \lambda_1$ is sufficiently large to create a point of minimum polar distance along the trajectory that is not one of the terminal points.

B. Launching Direction

So far we have written the bundle of all geodesics through (λ_1, ϕ_1) in the format (61), which specifies the change of latitude as a function of distance traveled. The direction at point 1 in the topocentric system of coordinates is represented by c_3 . To address the inverse problem of geodesy, that is to pick the particular geodesics which also runs through the terminal point (λ_2, ϕ_2) in fulfillment of the Dirichlet boundary conditions, the associated change in longitude, some form of (40),

$$\frac{d\lambda}{ds} = c_3 \frac{1}{(N+h)^2(1-\tau^2)} = \frac{c_3}{E}, \quad (63)$$

must obviously get involved. The strategy is to write down $\tau(\lambda)$ or $\lambda(\tau)$ with c_3 as a parameter, then to adjust c_3 to ensure with what would be called a shooting method that starting at point 1 at that angle eventually passes through point 2. Coupling of λ to τ is done by division of (61) and (63),

$$\frac{d\tau}{d\lambda} = \frac{d\tau}{ds} \frac{ds}{d\lambda} = \frac{d\tau}{ds} \frac{d\lambda}{ds} = \pm \frac{(N+h)^2(1-\tau^2) \sqrt{1 - \tau^2 - \frac{c_3^2}{(N+h)^2}}}{c_3(h+M)}. \quad (64)$$

Separation of both variables yields

$$\pm \int_1 d\tau \frac{c_3(h+M)}{(N+h)^2(1-\tau^2)\sqrt{1-\tau^2-\frac{c_3^2}{(N+h)^2}}} = \lambda_{|1}. \quad (65)$$

For short paths, small $|\lambda_2 - \lambda_1|$, the integral is simply $\int_{\tau_1}^{\tau_2}$. If one of the cases occurs, where the difference in λ is too large for a solution, the additional path through the singularity τ_m is to be used [with some local extremum in the graph $\tau(s)$], and the integral is to be interpreted as $\int_{\tau_1}^{\tau_m} - \int_{\tau_m}^{\tau_2}$. Taking the sign change and the symmetry of the integrand into account, this is $\int_{\tau_1}^{\tau_2} + 2 \int_{\tau_2}^{\tau_m} = \int_{\tau_1}^{\tau_m} + \int_{\tau_2}^{\tau_m}$, twice the underivative at τ_m minus the sum of the underivative at τ_1 and τ_2 . In the right plot of Figure 3, the difference is in including or not including the loudspeaker shaped area between τ_2 and τ_m .

τ_m is the solution of (62), the positive value of the solution taken if $d\tau/ds > 0$ at ϕ_1 , the negative value if $d\tau/ds < 0$ at ϕ_1 . The squared zero of $d\tau/ds$, τ_m^2 , is a root of the fourth-order polynomial which emerges by rewriting (62),

$$\begin{aligned} e^4 h^4 (\tau_m^2)^4 + 2e^2 h^2 [\rho_e^2 - h^2 + e^2 (c_3^2 - h^2)] (\tau_m^2)^3 \\ + \left[(\rho_e^2 - h^2)^2 + 2e^2 (2h^4 - 2c_3^2 h^2 - 2\rho_e^2 h^2 - \rho_e^2 c_3^2) + e^4 (c_3^2 - h^2)^2 \right] (\tau_m^2)^2 \\ + 2 \left[-(\rho_e^2 - h^2)^2 + c_3^2 (\rho_e^2 + h^2) + e^2 \left(-(c_3^2 - h^2)^2 + \rho_e^2 (c_3^2 + h^2) \right) \right] \tau_m^2 \\ + [(\rho_e + h)^2 - c_3^2] [(\rho_e - h)^2 - c_3^2] = 0. \end{aligned} \quad (66)$$

Alternatively, τ_m admits a Taylor expansion by expanding the zero of (62) in orders of e :

$$\pm \tau_m = \sqrt{1 - \frac{c_3^2}{H^2}} + \frac{c_3^2 \rho_e}{2H^3} \sqrt{1 - \frac{c_3^2}{H^2}} e^2 - \frac{3c_3^2 \rho_e}{8H^3} \left[\left(\frac{\rho_e}{H} - \frac{c_3^2}{H^2} \right)^2 - 1 \right] \sqrt{1 - \frac{c_3^2}{H^2}} e^4 + O(e^6), \quad (67)$$

where some maximum distance

$$H \equiv \rho_e + h \quad (68)$$

to the polar axis has been defined to condense the notation.

The integrand in (65) has a Taylor expansion in e ,

$$\pm \int d\tau \left[\frac{c_3/(\rho_e + h)}{(1-\tau^2)\sqrt{1-\tau^2-\frac{c_3^2}{(\rho_e+h)^2}}} - \frac{c_3 \rho_e [(\rho_e + h)^2(2-\tau^2) - 2c_3^2]}{2(\rho_e + h)^4 \left(1-\tau^2-\frac{c_3^2}{(\rho_e+h)^2}\right)^{3/2}} e^2 + O(e^4) \right] = \lambda_{|1}. \quad (69)$$

We integrate the left hand side of (69) separately for each power of e up to $O(e^4)$,

$$\begin{aligned} \left[\arctan \frac{\tau \frac{c_3}{H}}{\sqrt{T}} - \frac{c_3 \rho_e}{2H^2} \left(\frac{\tau}{\sqrt{T}} + \arctan \frac{\tau}{\sqrt{T}} \right) e^2 \right. \\ \left. - \frac{c_3 \rho_e}{16H^7} \left(\frac{\tau}{T^{3/2}} [-2H^2 c_3^2 \rho_e + 4H^2 c_3^2 \rho_e T + 2c_3^4 \rho_e - 6H^3 c_3^2 T + 6H^5 T - 9H^5 T^2 - 4H^4 \rho_e T + 6H^4 \rho_e T^2] \right. \right. \\ \left. \left. + H^2 [3H^3 - 2\rho_e H^2 - 3c_3^2 H + 4c_3^2 \rho_e] \arctan \frac{\tau}{\sqrt{T}} \right) e^4 + O(e^6) \right]_{\tau_1}^{\tau_2} = \pm (\lambda_2 - \lambda_1), \end{aligned} \quad (70)$$

where

$$T \equiv 1 - \tau^2 - \left(\frac{c_3}{H} \right)^2 \quad (71)$$

is some convenient definition of the trajectory's distance to its solstice τ_m . The first term is not the principal value of the arc tangent but its steadily defined extension through the entire interval of τ -values, $|\tau| \leq \tau_m$. It is an odd function of τ , and phase jumps are corrected as follows: the inclination at $\tau = 0$ has (by inspection of the derivative above at $\tau = 0$) the sign of c_3 . So whenever the triple product $\text{sgn } \tau \text{sgn } c_3 \text{sgn}(\arctan \cdot)$ is negative, one must shift the branch of the arctan by adding multiples of $\pi \text{sgn } \tau \text{sgn } c_3$ to the principal value.

Whether this is to be taken between the limits τ_2 and τ_1 or as the sum of two components (see above) can be tested by integrating up to τ_m , $\int_{\tau_1}^{\tau_m}$, where the value of the underivative at τ_m is given by $(1/2) \arctan 0$, effectively $\frac{\pi}{2} \text{sgn } \tau_m \text{sgn } c_3$ after selecting the branch of the inverse trigonometric function as described above.

Equation (70) is solved numerically, where c_3/H is the unknown, where $\tau_{1,2}$ and $\lambda_{1,2}$ are known from the coordinates of the two points that define the boundary value problem, and where e and H are constant parameters. The complexity of the equation suggests use of a Newton algorithm to search for the zero, starting from (A19), $c_3/H \approx \cos \phi_1 \cos \phi_2 \sin(\lambda_2 - \lambda_1)/\sin Z$, as the initial estimate.

An alternative is to insert the power series

$$c_3 = c_3^{(0)} + c_3^{(2)}e^2 + c_3^{(4)}e^4 + \dots \quad (72)$$

right into (65), integrate the orders of e term-by-term, and to obtain the coefficients $c_3^{(2)}$, $c_3^{(4)}$ etc. by comparison with the equivalent powers of the right hand side. $c_3^{(0)}$ is given by (A19); the coefficients of the higher powers are recursively calculated from linear equations. $c_3^{(2)}$, for example, is determined via

$$\left[\frac{\rho_e c_3^{(0)}}{H^2} \sqrt{1 - \tau^2 - \left(\frac{c_3^{(0)}}{H}\right)^2} \left[1 - \frac{c_3^{(0)^2}}{H^2} \right] \arctan \frac{H\tau}{\sqrt{1 - \tau^2 - \left(\frac{c_3^{(0)}}{H}\right)^2}} + \frac{\rho_e c_3^{(0)}}{H^2} \left[\frac{1 - c_3^{(0)^2}}{H^2} \right] \tau - 2 \frac{c_3^{(2)}}{H} \tau \right]_{\tau_1}^{\tau_2} = 0. \quad (73)$$

The corresponding equation for $c_3^{(4)}$ is already too lengthy to be reproduced here, so no real advantage remains in comparison with solving the non-linear equation (69).

Once the parameter c_3 is known for a particular set of terminal coordinates, $\lambda(\phi)$ is given by replacing τ_2 and λ_2 in (70) by any other generic pair of values. Two other variables of interest along the curve, the direction and integrated distance from the starting point, are then accessible with methods summarized in the next, final two sub-chapters.

C. Nautical course

The nautical course at any point of the trajectory $\phi(\lambda)$ is the angle κ in the topocentric coordinate system spanned by \mathbf{e}_ϕ (direction North) and \mathbf{e}_λ (direction East), measured North over East,

$$\frac{d\mathbf{r}}{ds} = \cos \kappa \frac{\mathbf{e}_\phi}{|\mathbf{e}_\phi|} + \sin \kappa \frac{\mathbf{e}_\lambda}{|\mathbf{e}_\lambda|} = \sin \kappa \frac{\mathbf{e}_\lambda}{\sqrt{E}} + \cos \kappa \frac{\mathbf{e}_\phi}{\sqrt{G}}. \quad (74)$$

$d\mathbf{r}/ds$ is the differential of (15) with respect to s , where $d/ds = (d\lambda/ds)(d/d\lambda) + (d\phi/ds)(d/d\phi)$,

$$\frac{d\mathbf{r}}{ds} \propto \frac{d\lambda}{ds} \mathbf{e}_\lambda + \frac{d\phi}{ds} \mathbf{e}_\phi. \quad (75)$$

The sign \propto indicates that the left hand side of this equation is a vector normalized to unity, but not the right hand side.

$$\frac{\sin \kappa / \sqrt{E}}{\cos \kappa / \sqrt{G}} = \frac{d\lambda/ds}{d\phi/ds} = \frac{d\lambda}{d\phi}. \quad (76)$$

Insertion of (19), (21) and (44) yield

$$\frac{M+h}{(N+h)\cos\phi} \tan \kappa = \frac{1}{\frac{d\phi}{d\lambda}} = \frac{1}{\frac{d\phi}{d\tau} \frac{d\tau}{d\lambda}} = \frac{d\tau/d\phi}{\frac{d\tau}{d\lambda}} = \frac{\cos\phi}{\frac{d\tau}{d\lambda}}. \quad (77)$$

Mixing (64) into this yields the course, supposed ϕ and c_3 are known,

$$\tan \kappa = \pm \frac{c_3}{(N+h)\sqrt{\cos^2 \phi - \left(\frac{c_3}{N+h}\right)^2}}. \quad (78)$$

D. Distance To Terminal Points

An implicit write-up for the distance from the origin \mathbf{s}_1 measured along the curve is given by separating variables in (61):

$$s = \pm \int_{\tau_1} d\tau \frac{h(1 - e^2\tau^2) + N(1 - e^2)}{(1 - e^2\tau^2)\sqrt{1 - \tau^2 - \frac{c_3^2}{(N+h)^2}}}. \quad (79)$$

Power series expansion of the integrand in powers of e , then integration term-by-term, generate a Taylor series of the form

$$s = \pm(s^{(0)} + s^{(2)}e^2 + s^{(4)}e^4 + \dots)\Big|_{\tau_1}^{\tau}.$$

In the notation (71), the components of the underivative read:

$$s^{(0)} = H \arctan \frac{\tau}{\sqrt{1 - \tau^2 - c_3^2/H^2}} = H \arctan \frac{\tau}{\sqrt{T}}; \quad (80)$$

$$s^{(2)} = -\frac{\rho_e}{4} \left[(1 + c_3^2/H^2) \arctan \frac{\tau}{\sqrt{T}} + \tau \frac{3(1 - \tau^2) - c_3^2/H^2}{\sqrt{T}} \right]; \quad (81)$$

$$\begin{aligned} s^{(4)} = \frac{\rho_e}{64} \Big\{ & [9(c_3/H)^4 - 6(c_3/H)^2 - 12(c_3/H)^4(\rho_e/H) + 4(c_3/H)^2(\rho_e/H) - 3] \arctan \frac{\tau}{\sqrt{T}} \\ & + \frac{\tau}{T^{3/2}} \left[24T(c_3/H)^4 - 24T(c_3/H)^2 + 63T^2(c_3/H)^2 - 8(\rho_e/H)(c_3/H)^6 + 8(\rho_e/H)(c_3/H)^4 \right. \\ & \left. - 8(\rho_e/H)T(c_3/H)^4 + 8(\rho_e/H)T(c_3/H)^2 - 12(\rho_e/H)T^2(c_3/H)^2 - 27T^2 + 30T^3 \right] \Big\}. \quad (82) \end{aligned}$$

V. SUMMARY

Computation of the geodetic line within the iso-surface of constant altitude above the ellipsoid is of the same complexity as on its surface at zero altitude. Although the surface is no longer an ellipsoid, mathematics reaches an equivalent level of simplification at which one integral is commonly expanded in powers of the eccentricity or flattening factor. We have done this first for the parameter which provides a solution to the inverse problem, then for two of the basic functions, distance from the starting point and compass course.

APPENDIX A: REFERENCE: SPHERICAL CASE

The limit of vanishing eccentricity, $e = 0$, simplifies the curved trajectories to arcs of great circles, and presents an easily accessible first estimate of the series expansions in orders of e . The Cartesian coordinates of the two points to be connected then are

$$\mathbf{s}_1 = (\rho_e + h) \begin{pmatrix} \cos \phi_1 \cos \lambda_1 \\ \cos \phi_1 \sin \lambda_1 \\ \sin \phi_1 \end{pmatrix}; \quad \mathbf{s}_2 = (\rho_e + h) \begin{pmatrix} \cos \phi_2 \cos \lambda_2 \\ \cos \phi_2 \sin \lambda_2 \\ \sin \phi_2 \end{pmatrix}. \quad (A1)$$

The angular separation Z is derived from the dot product $\mathbf{s}_1 \cdot \mathbf{s}_2 = |\mathbf{s}_1||\mathbf{s}_2| \cos Z$,

$$\cos Z = \sin \phi_1 \sin \phi_2 + \cos \phi_1 \cos \phi_2 \cos(\lambda_1 - \lambda_2); \quad 0 \leq Z \leq \pi. \quad (A2)$$

Each point \mathbf{s} on the great circle in between lies in the plane defined by the sphere center and the terminal points, and can therefore be written as a linear combination

$$\mathbf{s} = \alpha \mathbf{s}_1 + \beta \mathbf{s}_2, \quad 0 \leq \alpha, \beta. \quad (A3)$$

The square must remain normalized to the squared radius

$$s^2 = \alpha^2 \mathbf{s}_1^2 + \beta^2 \mathbf{s}_2^2 + 2\alpha\beta \mathbf{s}_1 \cdot \mathbf{s}_2 = \alpha^2(\rho_e + h)^2 + \beta^2(\rho_e + h)^2 + 2\alpha\beta(\rho_e + h)^2 \cos Z, \quad (\text{A4})$$

which couples the two expansion coefficients via

$$\alpha^2 + \beta^2 + 2\alpha\beta \cos Z = 1. \quad (\text{A5})$$

One reduction to a single parameter ξ to enforce this condition is

$$\alpha = \frac{\cos \xi}{\sqrt{1 + \sin(2\xi) \cos Z}}; \quad \beta = \frac{\sin \xi}{\sqrt{1 + \sin(2\xi) \cos Z}}; \quad 0 \leq \xi \leq \pi/2. \quad (\text{A6})$$

In summary, the point (A3) on the great circle of radius $\rho_e + h$ between \mathbf{s}_1 and \mathbf{s}_2 has the Cartesian coordinates

$$\mathbf{s} = (\rho_e + h) \frac{\cos \xi}{\sqrt{1 + \sin(2\xi) \cos Z}} \begin{pmatrix} \cos \phi_1 \cos \lambda_1 \\ \cos \phi_1 \sin \lambda_1 \\ \sin \phi_1 \end{pmatrix} + (\rho_e + h) \frac{\sin \xi}{\sqrt{1 + \sin(2\xi) \cos Z}} \begin{pmatrix} \cos \phi_2 \cos \lambda_2 \\ \cos \phi_2 \sin \lambda_2 \\ \sin \phi_2 \end{pmatrix} \quad (\text{A7})$$

$$\equiv (\rho_e + h) \begin{pmatrix} \cos \phi(\xi) \cos \lambda(\xi) \\ \cos \phi(\xi) \sin \lambda(\xi) \\ \sin \phi(\xi) \end{pmatrix}. \quad (\text{A8})$$

The z -component of this,

$$\sin \phi(\xi) = \frac{\cos \xi \sin \phi_1 + \sin \xi \sin \phi_2}{\sqrt{1 + \sin(2\xi) \cos Z}}, \quad (\text{A9})$$

allows to convert ξ into φ . No ambiguity with respect to the branch of the arcsin arises since $-\pi/2 \leq \phi \leq \pi/2$. The ratio of the y and x -components demonstrates the dependence of λ on ξ ,

$$\tan \lambda(\xi) = \frac{\cos \xi \cos \phi_1 \sin \lambda_1 + \sin \xi \cos \phi_2 \sin \lambda_2}{\cos \xi \cos \phi_1 \cos \lambda_1 + \sin \xi \cos \phi_2 \cos \lambda_2}, \quad (\text{A10})$$

where the arctan branch is defined by considering separately the numerator and denominator under the restrictions that no signs are canceled. The azimuth angle σ between the points at $\xi = 0$ and at ξ follows from $\mathbf{s} \cdot \mathbf{s}_1 = (\rho_e + h)^2 \cos \sigma$,

$$\begin{aligned} \cos \sigma &= \left(\frac{\cos \xi}{\sqrt{1 + \sin(2\xi) \cos Z}} \begin{pmatrix} \cos \phi_1 \cos \lambda_1 \\ \cos \phi_1 \sin \lambda_1 \\ \sin \phi_1 \end{pmatrix} + \frac{\sin \xi}{\sqrt{1 + \sin(2\xi) \cos Z}} \begin{pmatrix} \cos \phi_2 \cos \lambda_2 \\ \cos \phi_2 \sin \lambda_2 \\ \sin \phi_2 \end{pmatrix} \right) \cdot \begin{pmatrix} \cos \phi_1 \cos \lambda_1 \\ \cos \phi_1 \sin \lambda_1 \\ \sin \phi_1 \end{pmatrix} \\ &= \frac{\sin \xi \cos Z + \cos \xi}{\sqrt{1 + \sin(2\xi) \cos Z}}; \quad 0 \leq \sigma \leq Z. \end{aligned} \quad (\text{A11})$$

The path length along the great circle perimeter is simply the radial distance $\rho_e + h$ to the center of coordinates times the azimuth angle σ measured in radians,

$$s = \int_1 \sqrt{ds^2} = (\rho_e + h)\sigma = (\rho_e + h) \arccos \frac{\sin \xi \cos Z + \cos \xi}{\sqrt{1 + \sin(2\xi) \cos Z}}; \quad 0 \leq s \leq (\rho_e + h)Z. \quad (\text{A12})$$

To calculate the direction in the \mathbf{e}_λ - \mathbf{e}_ϕ -plane at the starting point \mathbf{s}_1 , we employ ξ as the parameter that mediates between λ and ϕ :

$$\frac{d\lambda}{ds} = \frac{1}{\rho_e + h} \frac{d\lambda}{d\sigma} = \frac{1}{\rho_e + h} \frac{d\lambda}{d\xi} \frac{d\xi}{d\sigma} = \frac{1}{\rho_e + h} \frac{d\lambda}{d\xi} / \frac{d\sigma}{d\xi}. \quad (\text{A13})$$

To calculate $d\lambda/d\xi$, use the derivative of (A10) with respect to ξ ,

$$\begin{aligned} \frac{1}{\cos^2 \lambda} \frac{d\lambda}{d\xi} &= \frac{-\sin \xi \cos \phi_1 \sin \lambda_1 + \cos \xi \cos \phi_2 \sin \lambda_2}{\cos \xi \cos \phi_1 \cos \lambda_1 + \sin \xi \cos \phi_2 \cos \lambda_2} \\ &\quad - (\cos \xi \cos \phi_1 \sin \lambda_1 + \sin \xi \cos \phi_2 \sin \lambda_2) \frac{-\sin \xi \cos \phi_1 \cos \lambda_1 + \cos \xi \cos \phi_2 \cos \lambda_2}{(\cos \xi \cos \phi_1 \cos \lambda_1 + \sin \xi \cos \phi_2 \cos \lambda_2)^2}. \end{aligned} \quad (\text{A14})$$

In particular at the starting point, where $\xi = 0$, $\lambda = \lambda_1$ and $\phi = \phi_1$,

$$\frac{1}{\cos^2 \lambda_1} \frac{d\lambda}{d\xi}_{|1} = \frac{\cos \phi_2 \sin \lambda_2}{\cos \phi_1 \cos \lambda_1} - \cos \phi_1 \sin \lambda_1 \frac{\cos \phi_2 \cos \lambda_2}{(\cos \phi_1 \cos \lambda_1)^2}.$$

By multiplication with $\cos \phi_1 \cos^2 \lambda_1$

$$\cos \phi_1 \frac{d\lambda}{d\xi}_{|1} = \cos \phi_2 \sin \lambda_2 \cos \lambda_1 - \sin \lambda_1 \cos \phi_2 \cos \lambda_2 = \cos \phi_2 \sin(\lambda_2 - \lambda_1). \quad (\text{A15})$$

To calculate $d\sigma/d\xi$, we convert the cosine in (A11) to the sine,

$$\sin \sigma = \sqrt{1 - \cos^2 \sigma} = \frac{\sin \xi \sin Z}{\sqrt{1 + \sin(2\xi) \cos Z}}. \quad (\text{A16})$$

The derivative of this with respect to ξ is

$$\cos \sigma \frac{d\sigma}{d\xi} = \cos \xi \sin Z \frac{1}{\sqrt{1 + \sin(2\xi) \cos Z}} - \frac{1}{2} \sin \xi \sin Z \frac{2 \cos(2\xi) \cos Z}{(1 + \sin(2\xi) \cos Z)^{3/2}},$$

in particular at the starting point, $\sigma = \xi = 0$,

$$\frac{d\sigma}{d\xi}_{|1} = \sin Z. \quad (\text{A17})$$

Insert this derivative and (A15) back into (A13)

$$\frac{d\lambda}{ds}_{|1} = \frac{1}{\rho_e + h} \frac{\cos \phi_2 \sin(\lambda_2 - \lambda_1)}{\cos \phi_1} \frac{1}{\sin Z}, \quad (\text{A18})$$

to obtain the master parameter c_3 for the spherical case with (41),

$$c_3^{(0)} = (\rho_e + h) \frac{\cos \phi_1 \cos \phi_2 \sin(\lambda_2 - \lambda_1)}{\sin Z}; \quad (e = 0). \quad (\text{A19})$$

APPENDIX B: ZERO ALTITUDE

In particular *on* the ellipsoid, at $h = 0$, (formal) solutions to some integrals exist. An underivative of (79) in terms of Elliptic Integrals of the Second Kind is [8, 3.158.11][4, 219.07],

$$s = \pm \rho_e \left[\sqrt{1 - \frac{c_3^2 e^2}{\rho_e^2}} E \left(\arcsin \left(\frac{e\tau}{\sin \zeta} \right) \setminus \zeta \right) - e^2 \tau \sqrt{\frac{1 - e^2}{1 - e^2 \tau^2} - \frac{c_3^2}{\rho_e^2}} \right] + const; \quad h = 0, \quad (\text{B1})$$

where an auxiliary modular angle ζ is defined from

$$\sin \zeta \equiv e \sqrt{\frac{1 - c_3^2 / \rho_e^2}{1 - c_3^2 e^2 / \rho_e^2}}.$$

The left hand side of (65) can be written as an Elliptic Integral of the Third Kind [8, 3.157.7],

$$\pm \frac{c_3}{\rho_e} \frac{1 - e^2}{\sqrt{1 - c_3^2 e^2 / \rho_e^2}} \Pi \left(\frac{\sin^2 \zeta}{e^2}; \arcsin \left(\frac{e\tau}{\sin \zeta} \right) \setminus \zeta \right) + const = \lambda; \quad h = 0. \quad (\text{B2})$$

APPENDIX C: NOTATIONS

c_5, c_3	constants of integration
E, E_1	Gauss parameter eq. (19) and its value at curve origin (λ_1, ϕ_1, h)
$E(\cdot \setminus \cdot)$	Incomplete Elliptic Integral of the Second Kind in the Abramowitz-Stegun notation [1, §17]
e	eccentricity of the ellipsoid, eq. (1)
$\mathbf{e}_{\lambda, \phi}$	horizontal topocentric coordinate vectors at (λ, ϕ, h)
f	flattening, eq. (8)
F	Gauss parameter, eq. (20)
G, G_1	Gauss parameter, eq. (21) and its value at curve origin
$\Gamma_{\cdot\cdot}$	Christoffel symbols in (λ, ϕ, h) coordinates
h	vertical distance to surface of ellipsoid
H	maximum distance to polar axis (in the equatorial plane), eq. (68)
κ	nautical angle, North over East in the topocentric tangential plane
λ	longitude
M	a radius of curvature on the ellipsoid surface, eq. (18)
N	a radius of curvature on the ellipsoid surface, eq. (13)
ρ	distance ellipsoid center to foot point on the surface
$\rho_{e,p}$	equatorial, polar radius of ellipsoid, eq. (1)
s	distance along geodetic line measured from curve origin
\mathbf{s}	vector from ellipsoid center to point on geodetic line
S	length of curved geodetic trajectory; equals s_2 at (λ_2, ϕ_2, h)
sgn	sign function, ± 1 or 0
σ	azimuth along great circle, eq. (A12)
T	normalized distance from closest polar approach, eq (71)
τ, τ_1	$\sin \phi$ and its value at start of curve, eq. (43)
v	geodetic minus geocentric latitude
ϕ	geodetic latitude
ϕ'	geocentric latitude
$\Pi(\cdot; \cdot \setminus \cdot)$	Incomplete Elliptic Integral of the Third Kind [1, §17]
x, y, z	Cartesian coordinates from ellipsoid center, eq. (15)
ξ	parametrization of great circle (spherical case), eq. (A7)
Z	cone angle of circular section (spherical case), eq. (A2)

APPENDIX D: FEM IMPLEMENTATION

The simplest numerical solution of the inverse problem without restriction on the eccentricity could be a finite-element (FEM) integration of (65) and iterative adjustment of the single free parameter c_3 . The following Java program implements this approach. Member functions in overview: the constructors `Geod` define a surface from the parameters ρ_e , e and h in which the geodetic line is embedded. `getCartesian` computes the vector (15). `curvN` computes (13). `dNdtau` computes

$$\frac{dN}{d\tau} = N(\tau) \frac{e^2 \tau}{1 - e^2 \tau^2}. \quad (\text{D1})$$

`flatt` computes (8). `curvM` computes (18). `dMdtau` computes

$$\frac{dM}{d\tau} = 3M(\tau) \frac{e^2 \tau}{1 - e^2 \tau^2}. \quad (\text{D2})$$

`d2Mdtau2` computes

$$\frac{d^2 M}{d\tau^2} = 3M(\tau) \frac{e^2(1 + 4e^2 \tau^2)}{(1 - e^2 \tau^2)^2}. \quad (\text{D3})$$

`GaussE` computes (19). `dEdtau` computes

$$\frac{dE}{d\tau} = -2\tau(N + h)(M + h). \quad (\text{D4})$$

d2Edtau2 computes the next higher derivative, $d^2E/d\tau^2$. **GaussG** computes (21). **tauTropic** solves (66) for τ_m with a first estimate taken from (67); the function is not actually called in this implementation. **dtaudlambda** computes $d\tau/d\lambda$ via (64), referencing one factor to (61). **dsdlambda** is $ds/d\lambda$, the inverse of (63). **discrT** computes $1 - \tau^2 - c_3^2/(N+h)^2$, which generalizes (71) to nonzero e . Its derivatives

$$\frac{d}{d\tau} \left[1 - \tau^2 - \frac{c_3^2}{[N(\tau) + h]^2} \right] = 2\tau \left[\frac{Ne^2c_3^2}{(N+h)^3(1-e^2\tau^2)} - 1 \right]; \quad (D5)$$

$$\frac{d^2}{d\tau^2} \left[1 - \tau^2 - \frac{c_3^2}{[N(\tau) + h]^2} \right] = -2 \left[1 - \frac{Nc_3^2e^2}{(N+h)^3(1-e^2\tau^2)} \left(1 + \frac{3he^2\tau^2}{(N+h)(1-e^2\tau^2)} \right) \right], \quad (D6)$$

are implemented in **dTdttau** and **d2Tdttau2**. **dtauds** calculates (61). **d2taudlambda2** calculates the derivative of (64),

$$\frac{d^2\tau}{d\lambda^2} = \frac{d}{d\lambda} \frac{E(\tau)\sqrt{1-\tau^2-c_3^2/(N+h)^2}}{c_3(h+M)} = \frac{d\tau}{d\lambda} \frac{d}{d\tau} \frac{E(\tau)\sqrt{1-\tau^2-c_3^2/(N+h)^2}}{c_3(h+M)}. \quad (D7)$$

d3taudlambda3 is the next higher order application of Bruno di Faà's formula to relegate derivatives $d/d\lambda$ to derivatives $d/d\tau$, [8, 0.430],

$$\frac{d^3\tau}{d\lambda^3} = \frac{d^2\tau}{d\lambda^2} \frac{d}{d\tau} \frac{E(\tau)\sqrt{1-\tau^2-c_3^2/(N+h)^2}}{c_3(h+M)} + \left(\frac{d\tau}{d\lambda} \right)^2 \frac{d^2}{d\tau^2} \frac{E(\tau)\sqrt{1-\tau^2-c_3^2/(N+h)^2}}{c_3(h+M)}. \quad (D8)$$

ds2dlambda2 calculates

$$\frac{d^2s}{d\lambda^2} = \frac{d}{d\lambda} \frac{E}{c_3} = \frac{1}{c_3} \frac{d\tau}{d\lambda} \frac{d}{d\tau} E. \quad (D9)$$

c3Sphere returns the estimate (A19). **dtaudsSignum** returns the sign of $d\tau/ds$ at τ_1 , obtained by considering the sign of the derivative of (A9) with respect to ξ . **adjLambdaEnd** modifies ϕ_2 modulo 2π to select the smallest value of $|\phi_2 - \phi_1|$. **nautAngle** computes κ from (78).

tauShoot walks along a geodetic line on a discrete mesh of width $\Delta\lambda$ by extrapolating

$$\tau_{\lambda+\Delta\lambda} \approx \tau_\lambda + \frac{d\tau}{d\lambda} \Delta\lambda + \frac{1}{2} \frac{d^2\tau}{d\lambda^2} (\Delta\lambda)^2 + \frac{1}{6} \frac{d^3\tau}{d\lambda^3} (\Delta\lambda)^3, \quad (D10)$$

initialized at λ_1, ϕ_1 , given c_3 . The equivalent formula is used to build up $s_{\lambda+\Delta\lambda}$, currently only implemented up to the order $(\Delta\lambda)^2$, since **d3sdlambda3** returns 0. **c3shoot** calls **tauShoot** four times to adjust c_3 such that the error by which ϕ_2 was missed—returned by **tauShoot**—is minimized. The first call assumes (A19), the second takes an arbitrary small offset, and the third and fourth estimates are from linear and quadratic interpolations in the earlier calls to zoom into a root of this error as a function of c_3 . The last of these runs tabulates the Cartesian coordinates (15), λ, ϕ, κ and s on a subgrid of the λ -mesh. **main** collects some adjustable parameters plus the pairs (λ_1, ϕ_1) and (λ_2, ϕ_2) , and calls **c3shoot** to solve the inverse problem of geodetics.

package org.nevec.rjm ;

```
/** Solution of the inverse problem of geodesy for biaxial ellipsoid.
 * The inverse problem of geodesy is solved given the parameters of
 * a biaxial ellipsoid (equatorial radius and eccentricity), a common
 * altitude of a surface above the ellipsoid, and pairs of geodetic
 * coordinates (latitude, longitude) for a starting and a final point
 * of a trajectory.
 * <p>
 *
 * The first order differential equation of the smooth geodetic latitude
 * as a function of longitude is solved by moving from one point to the
 * next on a finite grid of equidistant sampling points on the longitude,
 * using a predictor-only Newton approximation.
 * The parameter which defines the initial direction in the local
 * tangential plane is gathered by a shooting method with a starting
 * guess obtained from a spherical approximation and a fixed number of
 * iteration through the parameter space.
 *
 * @since 2008-04-12
```

```

* @see preprint <a href="http://arxiv.org/abs/0711.0642">Geodetic line at constant altitude above the Ellipsoid</a>,
* \protect\vrule widthOpt\protect\href{http://arXiv.org/abs/0711.0642}{arXiv:0711.0642} [math.MG]
* @author Richard J. Mathar
*/
class Geod
{
    /** The inverse flattening factor of the IERS 2003 convention.
    */
    static public final double IERS_TN32_FLAT= 298.25642 ;

    /** The inverse flattening factor of the IERS 1996 convention.
    */
    static public final double IERS_TN21_FLAT= 298.25642 ;

    /** The inverse flattening factor of the GRS80 model.
    */
    static public final double GRS80_FLAT= 298.257222101 ;

    /** The inverse flattening factor of the WGS84 model.
    */
    static final double WGS84_FLAT= 298.257223563 ;

    /** Equatorial radius of the IERS 2003 convention, meters.
    */
    static public final double IERS_TN32_RHO_E=6378136.6 ;

    /** Equatorial radius of the IERS 1996 convention, meters.
    */
    static public final double IERS_TN21_RHO_E=6378136.49 ;

    /** Equatorial radius of the GRS 1980 system, meters.
    */
    static public final double GRS80_RHO_E=6378137.0 ;

    /** Equatorial radius of the WGS84, meters.
    */
    static public final double WGS84_RHO_E=6378137.0 ;

    /** Three enumeration values to convert angular units on user input and output
    */
    interface AngleUnit {
        int RAD =0, DEG = 1, GON = 2;
    } ;

    /** Equatorial radius in meters.
    */
    double rho_equat ;

    /** Eccentricity.
    * For the Earth of the order of 0.08.
    */
    double eccen ;

    /** Altitude of the inverse problem above the ellipsoid.
    * Value equals 0 on the ellipsoid surface. May have both signs.
    * Measured in the same units as Geod::rho_equat.
    */
    double altit ;

    /** Default Constructor.
    * The variables are set to the defaults of the surface of the WGS84 geoid.
    */
    public Geod()
    {
        this(WGS84_RHO_E,Math.sqrt((2.0-1./WGS84_FLAT)/WGS84_FLAT),0.) ;
    }

    /** Constructor.
    * @param rho equatorial radius [m]
    * @param e eccentricity

```



```

* @param h altitude above the reference ellipsoid [m]
* @todo catch error conditions of e>=1, rho<0 or h< polar radius.
*/
public Geod(double rho, double e, double h)
{
    rho_equat = rho ;
    eccen = e ;
    altit = h ;
}

/** Convert geodetic to Cartesian coordinates.
* @param tau the sine of the geodetic latitude
* @param lambd the geodetic longitude [rad]
* @return the three components of the Cartesian coordinates
* in the same units as rho_equat and altit.
* @see eq (15) of the preprint
*/
public double[] getCartesian(final double tau, final double lambd)
{
    double[] cart = new double[3] ;
    final double N = curvN(tau) ;
    final double sinphi = Math.sqrt(1.-tau*tau) ;
    cart[0] = (N+altit)*sinphi*Math.cos(lambd) ;
    cart[1] = (N+altit)*sinphi*Math.sin(lambd) ;
    cart[2] = (N*(1.+eccen)*(1.-eccen)+altit)*tau ;
    return cart ;
}

/** Curvature parameter N [m]
* @param tau sine of the latitude
* @return N according to eq (13) of the preprint.
*/
public double curvN(final double tau)
{
    return rho_equat/Math.sqrt( (1.+eccen*tau)*(1.-eccen*tau) ) ;
}

/** Derivative d N/d tau [m].
* @param tau sine of the latitude
* @return the derivative of N(tau)
*/
double dNdtau(double tau)
{
    return curvN(tau)*eccen*eccen*tau/((1.+eccen*tau)*(1.-eccen*tau)) ;
}

/** Convert eccentricity to flattening factor.
* @param e the (first) eccentricity
* @return f = 1-sqrt(1-eccentricity^2).
*/
public static double flatt(final double e)
{
    final double e2 = e*e ;
    if ( e2 < 3.8e-3)
        /* a Taylor approximation up to O(e^14), relative accuracy to E-16. */
        return e2*(0.5+e2*(1./8.+e2*(1./16.*e2*(5./128.+e2*(7./256.+e2*(21./1024.+e2*33./2048.)))))) ;
    else
        return 1.-Math.sqrt(1.-e2) ;
}

/** Flattening factor.
* @return f = 1-sqrt(1-eccentricity^2).
*/
public double flatt()
{
    return flatt(eccen) ;
}

```

```

/** Equatorial radius of curvature M [m]
 * @param tau sine of the latitude
 * @return M according to eq (18) of the preprint.
 */
public double curvM(final double tau)
{
    return curvN(tau)*(1.-eccen*eccen)/((1.+eccen*tau)*(1.-eccen*tau)) ;
}

/** Derivative d M /d tau [m]
 * @param tau sine of the latitude
 * @return the derivative of curvM() with respect to the parameter tau.
 */
public double dMdtau(final double tau)
{
    return 3.*curvM(tau)*eccen*eccen*tau/((1.+eccen*tau)*(1.-eccen*tau)) ;
}

/** second Derivative d^2 M /d tau^2 [m]
 * @param tau sine of the latitude
 * @return the second derivative of curvM() with respect to the parameter tau.
 */
public double d2Mdtau2(final double tau)
{
    final double etau2 = eccen*eccen*tau*tau ;
    return 3.*curvM(tau)*Math.pow(eccen/(1.-etau2),2.)*(1.+4.*etau2) ;
}

/** Gauss parameter E [m^2].
 * @param tau sine of the latitude
 * @return E according to eq (19) of the preprint.
 */
double GaussE(final double tau)
{
    return Math.pow(curvN(tau)+altit,2.)*(1.-tau*tau) ;
}

/** Derivative d E/ d tau [m^2]
 * @param tau sine of the latitude
 * @return the derivative of GaussE() with respect to the parameter tau
 */
public double dEdtau(final double tau)
{
    return -2.*tau*(curvN(tau)+altit)*(curvM(tau)+altit) ;
}

/** Derivative d^2 E/ d tau^2 [m^2]
 * @param tau sine of the latitude
 * @return the 2nd derivative of GaussE() with respect to the parameter tau
 */
public double d2Edtau2(final double tau)
{
    /* Use the Leibniz rule applied to the first derivative, which was -2*tau*(N+h)(M+h)
    */
    final double Nh = curvN(tau)+altit ;
    final double Mh = curvM(tau)+altit ;
    return -2.*(Nh*Mh+tau*dNdtau(tau)*Mh+tau*Nh*dMdtau(tau)) ;
}

/** Gauss parameter G [m^2].
 * @param tau sine of the latitude
 * @return G according to eq (21) of the preprint.
 * @since 2008-04-22
 */
double GaussG(final double tau)
{
    return Math.pow(curvM(tau)+altit,2.) ;
}

```

```

/** Minimum polar distance.
 * @param c3 the constant c3 along one individual trajectory
 * @return the value of tau_m according to eq (66) of the preprint that could be reached if the trajectory were
 * closed over the surface. That is the maximum of |sin(phi)|, or the value
 * of |sin(phi)| where the derivative of sin(phi) with respect to the longitude
 * becomes zero.
 * @since 2008-04-22
 */
double tauTropic(final double c3)
{
    final double c3sqr = c3*c3 ;
    final double H = rho_equat+altit ;
    final double rhoebar = rho_equat/H ;
    final double esqr = eccen*eccen ;

    /* Initial estimate from Taylor expansion in orders of squared eccentricity
     * See eq (67) of the preprint
     */
    double tau2 = (1.-c3sqr/(H*H))*(1+c3sqr*rhoebar*esqr/(H*H)*
        (1.+0.25*(3.*(1.-c3sqr/(H*H))+rhoebar*(7.*c3sqr/(H*H)-3.))*esqr)
        ) ;

    final double hsqr = altit*altit ;
    final double rhosqr = rho_equat*rho_equat ;

    /* coefficients in the 4th order polynomial of tau^2 which is to be solved */
    double[] coeff = new double[5] ;
    coeff[0] = (Math.pow(rho_equat+altit,2.)-c3sqr)*(Math.pow(rho_equat+altit,2.)-c3sqr) ;
    coeff[1] = 2.*(-Math.pow(rhosqr-hsqr,2.)
        +c3sqr*(rhosqr+hsqr)
        +esqr*(-Math.pow(c3sqr-hsqr,2.)+rhosqr*(c3sqr+hsqr))
        ) ;
    coeff[2] = Math.pow(rhosqr-hsqr,2.)
        +2.*esqr*(2.*hsqr*hsqr-2.*c3sqr*hsqr-2.*rhosqr*hsqr-rhosqr*c3sqr)
        +esqr*esqr*Math.pow(c3sqr-hsqr,2.) ;
    coeff[3] = 2.* Math.pow(altit*eccen,2.)*(rhosqr-hsqr+esqr*(c3sqr-hsqr)) ;
    coeff[4] = Math.pow(altit*eccen,4.) ;

    /* Small number of loops for self-consistency, using some Horner scheme for
     * the function to be zeroed and its first derivative */
    for(int loop=0; loop < 4 ;loop++)
    {
        System.out.println("# taum2 "+tau2) ;
        tau2 -= ( coeff[0]+tau2*(coeff[1]+tau2*(coeff[2]+tau2*(coeff[3]+tau2*coeff[4])))
            )/(
                coeff[1]+tau2*(2.*coeff[2]+tau2*(3.*coeff[3]+tau2*4.*coeff[4]))
            ) ;
    }
    return Math.sqrt(tau2) ;
}

/** Derivative d tau/d lambda.
 * @param tau sine of the latitude
 * @param c3 the constant c3 along one individual trajectory
 * @return absolute value of the derivative according to eq (64) of the preprint.
 */
double dtaudlambda(double tau, double c3)
{
    /* treat the factors (N+h)^2*(1-tau^2) and the sqrt(..)/(h+M) separately */
    return GaussE(tau)*dtauds(tau,c3)/c3 ;
}

/** Derivative d s/d lambda.
 * @param tau sine of the latitude
 * @param c3 the constant c3 along one individual trajectory
 * @return absolute value of the derivative of the path length according to eq (63) of the preprint.
 */
double dsdlambda(double tau, double c3)
{

```

```

    return GaussE(tau)/c3 ;
}

/** Discriminant under the square root of d tau / ds.
 * @param tau sine of the latitude
 * @param c3 the constant c3 along one individual trajectory
 * @return 1-tau^2-(c3/(N+h))^2.
 */
double discrT(final double tau, final double c3)
{
    /* N+altitude */
    final double Nh = curvN(tau)+altit ;
    return 1.-tau*tau-Math.pow(c3/Nh,2.) ;
}

/** Derivative of T with respect to tau.
 * @param tau sine of the latitude
 * @param c3 the constant c3 along one individual trajectory
 * @return derivative of discrT() with respect to tau
 */
double dTdttau(final double tau, final double c3)
{
    final double N = curvN(tau) ;
    final double Nh = N+altit ;
    return 2*tau*( N*Math.pow(c3*eccen/Nh,2.)/(Nh*(1.+eccen*tau)*(1.-eccen*tau)) -1. ) ;
}

/** Second derivative of T with respect to tau.
 * @param tau sine of the latitude
 * @param c3 the constant c3 along one individual trajectory
 * @return the second derivative of discrT() with respect to tau
 */
double d2Tdttau2(final double tau, final double c3)
{
    final double N = curvN(tau) ;
    final double Nh = N+altit ;
    final double e2 = eccen*eccen ;
    final double Oneetau = (1.+eccen*tau)*(1.-eccen*tau) ;
    return -2.*( 1.-N*c3*c3*e2*(1.+3.*altit*e2*tau*tau/Nh/Oneetau)/Math.pow(Nh,3.)/Oneetau ) ;
}

/** Derivative d tau/d s [1/m].
 * @param tau sine of the latitude
 * @param c3 the constant c3 along one individual trajectory
 * @return absolute value of the derivative according to eq (61) of the preprint.
 */
double dtauds(double tau, double c3)
{
    return Math.sqrt( discrT(tau,c3) )/(altit+curvM(tau)) ;
}

/** Second derivative d^2 tau/d lambda^2.
 * @param tau sine of the latitude
 * @param c3 the constant c3 along one individual trajectory
 * @return value of the derivative of eq (64) of the preprint.
 */
double d2taudlambda2(double tau, double c3)
{
    /* The array contains values of E, sqrt(T) and 1/(h+M) in the components facto[][0],
     * and their first derivatives with respect to tau in facto[][1].
     */
    double[][] facto = new double[3][2] ;

    facto[0][0] = GaussE(tau) ;
    facto[1][0] = Math.sqrt(discrT(tau,c3) ) ;
    facto[2][0] = 1./( curvM(tau)+altit) ;

    facto[0][1] = dEdtau(tau) ;
    /* derivative of square root is one half of the inverse multiplied by interior derivative
     */

```

```

facto[1][1] = dTdttau(tau,c3)/(2.*facto[1][0]) ;
/* derivative of 1/(h+M) is negative of its square multiplied by interior derivative
*/
facto[2][1] = -dMdttau(tau)*facto[2][0]*facto[2][0] ;

/* The Leibniz rule of derivatives is applied to collect the derivative of
* the product of facto[i][0], i=0..2.
*/
final double resul = facto[0][1]*facto[1][0]*facto[2][0] +facto[0][0]*facto[1][1]*facto[2][0]
+facto[0][0]*facto[1][0]*facto[2][1] ;

/* resul contains so far the derivative (d/dtau) (d tau/d lambda), without
* the c3 in the denominator. Postmultiply with d tau/ d lambda to generate
* the d^2 tau/ d lambda^2.
*/
return dtaudlambda(tau,c3)*resul/c3 ;
}

/** Third derivative d^3 tau/d lambda^3.
* @param tau sine of the latitude
* @param c3 the constant c3 along one individual trajectory
* @return value of the 2nd derivative of eq (64) of the preprint.
*/
double d3taudlambda3(double tau, double c3)
{
    final double T = discrT(tau,c3) ;

    /* Faa di Bruno formula:
    * d^3 tau/d lambda^3 = (d^2tau/d lambda^2)* (d/dtau) (dtau/dlambda)+ (dtau/dlambda)^2* (d^2/dtau^2) (dtau/dlambda)
    */
    /* The array contains values of E, sqrt(T) and 1/(h+M) in the components [][0],
    * and their first derivatives with respect to tau in [][1], 2nd derivatives in [][2].
    */
    double[][] facto = new double[3][3] ;

    facto[0][0] = GaussE(tau) ;
    facto[1][0] = Math.sqrt(T) ;
    facto[2][0] = 1./ ( curvM(tau)+altit) ;

    facto[0][1] = dEdtau(tau) ;
    /* derivative of square root is one half of the inverse multiplied by interior derivative
    */
    final double Tprime = dTdttau(tau,c3) ;
    facto[1][1] = Tprime/(2.*facto[1][0]) ;

    /* derivative of 1/(h+M) is negative of its square multiplied by interior derivative
    */
    final double Mprime = dMdttau(tau) ;
    facto[2][1] = -Mprime*facto[2][0]*facto[2][0] ;

    facto[0][2] = d2Edtau2(tau) ;

    /* First derivative of sqrt(T) was dTdttau/(2*sqrt(T)). The 2nd derivative therefore is
    * [ (d^2/dtau^2) T -(d T/d tau)^2/(2T)]/[2sqrt(T)], which we store in facto[1][2].
    */
    facto[1][2] = ( d2Tdttau2(tau,c3) -Tprime*Tprime/( 2.*T ) )/(2.*facto[1][0]) ;

    /* Second derivative of 1/(h+M). First one was -(dM/dtau)/(h+M)^2. Second one is
    * -(d^2M/dtau^2)/(h+M)^2+2(dM/dtau)^2/(h+M)^3.
    */
    facto[2][2] = ( 2.*Mprime*Mprime*facto[2][0]-d2Mdttau2(tau) )*facto[2][0]*facto[2][0] ;

    /* The value of (d/d tau) (d tau/d lambda). The implementation is not
    * tuned for quick evaluation, since the same value is calculated again in #d2taudlambda2.
    */
    final double dtaudlambdadtau = ( facto[0][1]*facto[1][0]*facto[2][0] +facto[0][0]*facto[1][1]*facto[2][0]
    +facto[0][0]*facto[1][0]*facto[2][1] )/c3 ;

    /* the value of (d^2/dtau^2) (dtau/dlambda) with Leibniz rule, multinomial case
    */

```

```

final double d3taudlambdadttau2 = ( facto[0][2]*facto[1][0]*facto[2][0] +2.*facto[0][1]*facto[1][1]*facto[2][0]
+2.*facto[0][1]*facto[1][0]*facto[2][1] +facto[0][0]*facto[1][2]*facto[2][0]
+2.*facto[0][0]*facto[1][1]*facto[2][1] +facto[0][0]*facto[1][0]*facto[2][2] )/c3 ;

/* Faa di Bruno rule of derivatives is applied to collect the derivative
* (d^2/dlambda^2) dtau/dlambda
* = (d^2 tau/dlambda^2)*(d/dtau) (d tau/d lambda)+(d tau/d lambda)^2* (d^2/d tau^2) (d tau/d lambda)
*/
return d2taudlambda2(tau,c3)*dtaudlambdadttau + Math.pow(dtaudlambda(tau,c3),2.)* d3taudlambdadttau2;
}

/** Second derivative d^2 s/d lambda^2 = (d/d lambda) (E/c3).
* @param tau sine of the latitude
* @param c3 the constant c3 along one individual trajectory
* @return value of the 2nd derivative of eq (63) of the preprint.
*/
double d2sdlambda2(double tau, double c3)
{
    /* First the derivative (d/dtau) (d s/d lambda), without
    * the c3 in the denominator. Postmultiply with d tau/ d lambda to generate
    * the d^2 s/ d lambda^2.
    */
    return dtaudlambda(tau,c3)*dEdtau(tau)/c3 ;
}

/** Third derivative d^3 s /d lambda ^3.
* This is currently implemented as a dummy to return 0. Effectively
* the order of the approximation remains second order.
* @param tau sine of the latitude
* @param c3 the constant c3 along one individual trajectory
* @return value of the third derivative of eq (67) of the preprint
*/
double d3sdlambda3(double tau, double c3)
{
    return 0. ;
}

/** Spherical approximation to parameter c3
* @param phi the latitudes at start and end [rad]
* @param lambd the longitudes at start and end [rad]
* @return the parameter c3 obtained form the spherical approximation.
* @see eq (A19) in the preprint.
*/
double c3Sphere(final double[] phi, final double lambd[])
{
    /* cosine of the angle Z */
    final double cosZ = Math.sin(phi[0])*Math.sin(phi[1])
+Math.cos(phi[0])*Math.cos(phi[1])*Math.cos(lambd[1]-lambd[0]) ;

    return (altit+rho_equat)*Math.cos(phi[0])*Math.cos(phi[1])*Math.sin(lambd[1]-lambd[0])/Math.sqrt(1.-cosZ*cosZ) ;
}

/** Approximate sign d tau / ds
* @param phi the latitudes at start and end [rad]
* @param lambd the longitudes at start and end [rad]
* @return the sign of the square root in eq (64) of the preprint, as obtained from
* the spherical approximation.
*/
double dtaudsSignum(final double[] phi, final double lambd[])
{
    /* cosine of the angle Z */
    final double cosZ = Math.sin(phi[0])*Math.sin(phi[1])
+Math.cos(phi[0])*Math.cos(phi[1])*Math.cos(lambd[1]-lambd[0]) ;
    return Math.signum(Math.sin(phi[1])-Math.sin(phi[0])*cosZ) ;
}

/** Adjust longitude of the final point.
* @param lambd the longitudes at start and end [rad]
* The value of lambd[1] is adjusted to the range lambd[0]+-pi to ensure that the

```

```

* trajectory chooses a path along the correct hemisphere.
*/
static void adjLambdaEnd(double lambd[])
{
    final double dl = lambd[1]-lambd[0] ;
    if ( Math.abs( dl+2.*Math.PI ) < Math.abs(dl) )
        lambd[1] += 2.*Math.PI ;
    if ( Math.abs( dl-2.*Math.PI ) < Math.abs(dl) )
        lambd[1] -= 2.*Math.PI ;
}

/** Nautical course in the local oblique horizontal [rad]
* @param tau sine of the latitude
* @param c3 the directional parameter of the solution to the inverse problem [m]
* @param north sign of the square root, +1 or -1
* @return the direction angle (North over East) in the local oblique tangential plane [rad]
* @see eq (78) of the preprint
*/
double nautAngle(final double tau, final double c3, final double north)
{
    /* sinkappa and coskappa are the sine and cosine of the angle multiplied by
    * a common positive factor, which retains the correct quadrant of the solution.
    */
    final double sinkappa = c3/(curvN(tau)+altit) ;
    final double coskappa = north* Math.sqrt( discrT(tau,c3) ) ;
    return Math.atan2(sinkappa,coskappa) ;
}

/** Compute one trajectory over the lambda interval, assuming c3 given.
* @param phi start and (target) value of the geodetic latitude [rad]
* @param lambd start and end value of the longitude [rad]
* @param c3 fixed parameter of c3
* @param Nsampl number of steps into which the interval lambd[0]-lambd[1] is divided
* @param usampl if positive, each usampl'th point of the result is printed
* @param useRad one value of the AngleUnit to indicate which units are preferred for angles
* @param taylOrd the order of the Taylor approximation in the FEM, must be 2 or 3.
* @return the overshooting of the value of tau versus the sine of phi[1] [rad]
*/
double tauShoot(final double[] phi, final double lambd[], final double c3, final int Nsampl, final int usampl,
    final int useRad, final int taylOrd)
{
    /* step width in the longitudes, including a sign */
    final double dlam = (lambd[1]-lambd[0])/Nsampl ;

    /* current value of tau = sin(phi), initialized with the value at the start */
    double tau = Math.sin(phi[0]) ;

    /* current value of s, the length along the line, initialized with the value at the start */
    double s = 0. ;

    /* the sign which chooses a N or a S route. +1 or -1 */
    double north = dtaudsSignum(phi,lambd) ;

    if ( usampl > 0 )
        System.out.println("\n\n") ;

    double kappa = nautAngle(tau,c3,north) ;
    System.out.println("# start course "+ kappa +" rad, " + Math.toDegrees(kappa)+ " deg") ;

    /* loop over the finite elements */
    for(int i=0; i < Nsampl; i++)
    {
        /* in each of the elements, a 2nd order Taylor approximation
        * for dtau/ds is applied with (64) of the preprint
        */
        double deriv1 = north*dtaudlambda(tau,c3) ;

        /* below, the 'north' factor appears twice and cancels */

```

```

double deriv2 = d2taudlambda2(tau,c3) ;

double deriv3 = (tayl0rd >= 3) ? north*d3taudlambda3(tau,c3) : 0.;

/* in each of the elements, a 2nd order Taylor approximation
 * for ds/dlambda is applied with (64) of the preprint
 */
double deriv1s = dsdlambda(tau,c3) ;
double deriv2s = d2sdlambda2(tau,c3) ;
double deriv3s = (tayl0rd >= 3) ? d3sdlambda3(tau,c3) : 0.;

/* Taylor extrapolation. Current value +(df/dx)*delta+(1/2)(d^2/dx^2)*delta^2+(1/6)*(d^3/dx^3)*delta^3
 * Update to the value at the right end of the interval.
 */
tau += dlam*(deriv1+dlam*(0.5*deriv2+dlam*deriv3/6.) ) ;

/* We may have passed by a maximum of the equatorial distance
 * and sense this by comparing the location of the extremum on the current parabolic
 * estimator for tau(lambda) with the current interval. Setting the derivative of the
 * Taylor estimator to zero, (df/dx)+(d^2/dx^2)*(lambda-current lambda)=0, the
 * location of the extremum is -(df/dx)/(( d^2f/dx^2) relative to the current lambda.
 * @todo implement the 3rd order Taylor Approximation as well.
 */
final double lamFlat = -deriv1/deriv2 ;
/* Update of the path length along the trajectory in the Taylor approximation
 */
s += dlam*(deriv1s+dlam*(0.5*deriv2s+dlam*deriv3s/6.) ) ;

if ( usampl > 0 )
{
    if ( i % usampl == 0 || i == Nsampl-1 )
    {
        /* longitude at right end of interval [rad] */
        final double longi = lambd[0]+(i+1)*dlam ;
        double[] cart = getCartesian(tau, longi ) ;

        /* Print the x,y,z Cartesian coordinates as the first 3 columns per line */
        System.out.print(""+cart[0]+" "+cart[1]+" "+cart[2]) ;

        kappa = nautAngle(tau,c3,north) ;

        /* Print longitude, geodetic latitude, nautical angle as columns 4 to 6 */
        if ( useRad == AngleUnit.RAD )
            System.out.print(" "+ longi +" "+ Math.asin(tau)+ " "+kappa ) ;
        else if ( useRad == AngleUnit.DEG )
            System.out.print(" "+ Math.toDegrees(longi) + " "+ Math.toDegrees(Math.asin(tau))
                +" "+Math.toDegrees(kappa) ) ;
        else
            System.out.print(" "+ 10.*Math.toDegrees(longi)/9. + " "
                + 10.*Math.toDegrees(Math.asin(tau))/9.
                +" "+10.*Math.toDegrees(kappa)/9. ) ;

        /* Print length of trajectory in column 7 */
        System.out.print(" "+s) ;
        System.out.println("");
    }
}

if ( lamFlat/dlam >= 0. && lamFlat/dlam < 1.)
{
    north *= -1. ;
}
}

/* tau now is the estimate of the final position; return the error */
return tau-Math.sin(phi[1]) ;
}

/** Perform 4 iterations on adjusting the parameter c3.
 * @param phi the latitudes at start and end [rad]
 * @param lambd the longitudes at start and end [rad]

```



```

* @param Nsampl the number of divisions of the longitude axis
* @param usampl undersampling factor for the printout in the converged trajectory
* @param useRad the unit of the angles which are reported
* @param tayl0rd the order of the Taylor expansion of the core integral, 2 or 3.
* @return a convergent to the parameter c3
*/
public double c3shoot(final double[] phi, final double lambd[], final int Nsampl, final int usampl, final int useRad,
    final int tayl0rd)
{
    /* For a parabolic (2nd order) approximation of the shooting
    * error (units of tau) use 3 abscissa and target missing parameters
    */
    double[] err = new double[4] ;
    double[] c3spread = new double [4] ;

    /* start with the estimate from the spherical approximation, zero eccentricity */
    c3spread[0] = c3Sphere(phi,lambd) ;
    err[0] = tauShoot(phi,lambd,c3spread[0],Nsampl,-1,useRad,tayl0rd) ;

    System.out.println("# c3 (spherical) "+c3spread[0]+" err "+err[0]) ;

    /* A stepping parameter for the variation */
    final double c3Delt = -c3spread[0]*0.005 ;

    c3spread[1] = c3spread[0]+c3Delt ;
    err[1] = tauShoot(phi,lambd,c3spread[1],Nsampl,-1,useRad,tayl0rd) ;

    /* estimate of the 3rd c3 by a linear interpolation
    * between the previous 2 to get zero error. The ansatz is err=err[0]+(err[1]-err[0])/c3Delt*(c-c[0]) = 0,
    * where (err[1]-err[0])/c3Delt is the slope of the error curve, which is solved for c.
    */
    c3spread[2] = c3spread[0]-c3Delt*err[0]/(err[1]-err[0]) ;
    err[2] = tauShoot(phi,lambd,c3spread[2],Nsampl,-1,useRad,tayl0rd) ;
    System.out.println("# c3 (linear extrapol) "+c3spread[2]+" err "+err[2]) ;

    /* Lagrange quadratic interpolation between (c3spread[0],err[0]), (c3spread[1],err[1])
    * and (c3spread[2],err[2]): err = eofCinter[0]+eofCinter[1]*c3spread+eofCinter[2]*c3spread^2
    * up to a common factor which is omitted.
    */
    double[] eofCinter = new double [3] ;
    eofCinter[0] = err[2]*c3spread[0]*c3spread[1]*(c3spread[0]-c3spread[1])
        +err[1]*c3spread[0]*c3spread[2]*(c3spread[2]-c3spread[0])
        +err[0]*c3spread[1]*c3spread[2]*(c3spread[1]-c3spread[2]) ;
    eofCinter[1] = err[2]*(Math.pow(c3spread[1],2.)-Math.pow(c3spread[0],2.))
        + err[1]*(Math.pow(c3spread[0],2.)-Math.pow(c3spread[2],2.))
        + err[0]*(Math.pow(c3spread[2],2.)-Math.pow(c3spread[1],2.)) ;
    eofCinter[2] = err[2]*(c3spread[0]-c3spread[1])
        +err[1]*(c3spread[2]-c3spread[0])
        +err[0]*(c3spread[1]-c3spread[2]) ;

    /* Set this quadratic to zero; normalize to monic polynomial */
    eofCinter[0] /= eofCinter[2] ;
    eofCinter[1] /= eofCinter[2] ;
    eofCinter[2] = Math.sqrt(0.25*Math.pow(eofCinter[1],2)-eofCinter[0]) ;

    /* From the two solutions take the one closer to the previous estimate */
    c3spread[3] = (c3spread[2]+0.5*eofCinter[1] > 0.) ? -0.5*eofCinter[1]+eofCinter[2] : -0.5*eofCinter[1]-eofCinter[2] ;
    err[3] = tauShoot(phi,lambd,c3spread[3],Nsampl,usampl,useRad,tayl0rd) ;
    System.out.println("# c3 (quadratic extrapol) "+c3spread[3]+" err "+err[3]) ;

    return c3spread[3] ;
}

/**
 * usage:
 * <tt>
 * java Geod [-h altitude/m] [-s #longitude steps] [-u sfac] [-R equat radius/m] [-e eccent] [-r]
 *          [-g] [-2] phi1 lam1 phi2 lam2

```

```

* </tt>
* <br>
* Command line parameters
* <ul>
* <li>
* -h followed by a floating point number specifies the altitude of the
*   trajectory above the ellipsoid. This is given in the same units as those of the
*   parameter of the equatorial radius. The default is zero.
* </li>
* <li>
* -s followed by an integer denotes the number of steps between the starting and
*   final point of the trajectory in the finite element method. The default is 400.
* </li>
* <li>
* -u followed by an integer specifies an undersampling factor for listing the
*   parameters of the trajectory on standard output. The elements actually listed
*   are given by the sampling steps divided by the undersampling factor.
*   The default is 10.
* </li>
* <li>
* -R followed by a floating point number defines the equatorial radius of
*   the surface of the ellipsoid in the same units as the option for the altitude.
*   The default is the WGS84 value.
* </li>
* <li>
* -e followed by a floating point number defines the eccentricity of the
*   surface of the ellipsoid. The default is the WGS84 value.
* </li>
* <li>
* -r without any additional number specifies that the four mandatory command
*   line parameters are provided in units of radians. The default is degrees.
* </li>
* <li>
* -g without any additional number specifies that the four mandatory command
*   line parameters are provided in units of gons. The default is degrees.
* </li>
* </ul>
* There are four mandatory parameters, the geodetic latitude and longitude
* of the start of the trajectory, and the geodetic latitude and longitude of
* the final point of the trajectory.
* <br>
* The standard output contains comment lines (starting with hashes) and
* one line per point on the trajectory with the following columns (separated by blanks):
* <ul>
* <li>
* The first three columns are the x, y and z Cartesian coordinate of the point in the
* same units as with the equatorial radius and altitude parameters.
* </li>
* <li>
* Columns 4 and 5 are the longitude and geodetic latitude of the point
* in the same units (deg or rad) as the command line parameters.
* </li>
* <li>
* Column 6 is the nautical angle in the local topocentric tangential plane
* in the same units (deg or rad or gon) as the command line parameters.
* </li>
* <li>
* Column 7 is the distance along the geodetic
* in the same units as the command line parameters.
* </li>
* </ul>
* <br>
* Examples
* <ul>
* <li>
* <tt>
* java Geod -s 1000 -e 0.08227185417541244347812117 -R 6378206.4 8.973611111 -79.573333333333 21.435000000 -158.02583333
* </tt>
* computes the trajectory for the case from Panama to Hawaii by Paul D. Thomas,
* J. Geophys. Res. vol 70 (1965) p 3331 in his Figure 3.

```

```

* <li>
* <tt>
* java Geod -R 637838.799919243 -g -e 0.0815815883368028 54.262654 0 11.977469 115.959876
* </tt>
* computes the trajectory for the case from Paris to Saigon by H. M. Dufour
* in Bull. Geo. vol 32 (1953) p 26, with  $f=1/300$ , angles in gons.
* </ul>
* @since 2008-04-12
*/
public static void main(String[] args) throws Exception
{
    /* default equatorial radius set to WGS84 */
    double rho = WGS84_RHO_E ;

    /* default eccentricity set to WGS84
    * that is 0.08195646813308029492885488 ;
    */
    double e = (2.-1./WGS84_FLAT)/WGS84_FLAT ;

    /* default altitude above ellipsoid is zero */
    double h = 0. ;

    /* default number of elements */
    int Nsampl = 400 ;

    /* Default undersampling factor for the printed output.
    */
    int usampl = 10 ;

    /* Default Taylor order in the FEM is 3.
    */
    int taylOrd = 3 ;

    /* no defaults for the coordinates */
    double[] phi = new double[2] ;
    double[] lambd = new double [2] ;

    /* units of final angular arguments. */
    int useRad = AngleUnit.DEG ;

    /* parse options and parameters
    */
    for(int argc =0 ; argc < args.length; argc++)
    {
        if ( args[argc].compareTo("-h") == 0)
            h = Double.parseDouble(args[++argc]) ;
        else if ( args[argc].compareTo("-s") == 0 )
            Nsampl = Integer.parseInt(args[++argc]) ;
        else if ( args[argc].compareTo("-u") == 0 )
            usampl = Integer.parseInt(args[++argc]) ;
        else if ( args[argc].compareTo("-R") ==0 )
            rho = Double.parseDouble(args[++argc]) ;
        else if ( args[argc].compareTo("-e") ==0 )
            e = Double.parseDouble(args[++argc]) ;
        else if ( args[argc].compareTo("-r") ==0 )
            useRad = AngleUnit.RAD ;
        else if ( args[argc].compareTo("-g") ==0 )
            useRad = AngleUnit.GON ;
        else if ( args[argc].compareTo("-2") == 0 )
            taylOrd = 2 ;
        else
        {
            phi[0] = Double.parseDouble(args[argc++]) ;
            lambd[0] = Double.parseDouble(args[argc++]) ;
            phi[1] = Double.parseDouble(args[argc++]) ;
            lambd[1] = Double.parseDouble(args[argc++]) ;
        }
    }

    /* If user provided degrees or gons on the command line, convert to radians for further processing

```

```

* 400 gons are 360 degrees.
*/
switch( useRad)
{
case AngleUnit.DEG :
    for(int i=0 ; i < 2 ; i++)
    {
        phi[i] = Math.toRadians(phi[i]) ;
        lambd[i] = Math.toRadians(lambd[i]) ;
    }
    break ;
case AngleUnit.GON :
    for(int i=0 ; i < 2 ; i++)
    {
        phi[i] = Math.toRadians(360.*phi[i]/400.) ;
        lambd[i] = Math.toRadians(360.*lambd[i]/400.) ;
    }
    break ;
}

/* normalize input such that the final longitude is on the same hemisphere */
adjLambdaEnd(lambd) ;

/* log the input parameters */
System.out.println("# equat Radius "+ rho+", eccentricity "+e+", altitude "+h) ;
System.out.println("# inverse flattening "+ 1./(1.-Math.sqrt(1.-e*e)) ) ;
System.out.println("# start "+phi[0] + " rad, "+lambd[0]+" rad; end "+phi[1] + " rad, "+lambd[1]+" rad") ;
switch( useRad)
{
case AngleUnit.DEG :
    System.out.println("# start "+Math.toDegrees(phi[0]) + " deg, "+ Math.toDegrees(lambd[0])
        +" deg; end "+Math.toDegrees(phi[1]) + " deg, "+Math.toDegrees(lambd[1])+" deg") ;
    break ;
case AngleUnit.GON :
    System.out.println("# start "+10.*Math.toDegrees(phi[0])/9. + " gon, "+ 10.*Math.toDegrees(lambd[0])/9.
        +" gon; end "+10.*Math.toDegrees(phi[1])/9. + " gon, "+10.*Math.toDegrees(lambd[1])/9.+" gon") ;
}
System.out.println("# "+Nsampl + " elements, Taylor order " + taylOrd) ;

/* define the shape of the ellipsoid */
Geod g = new Geod(rho,e,h) ;

/* Determine the main parameter of the bundle of geodesics through initial point to
* solve the inverse problem, and print parameters along the trajectory
* in the 4th iteration.
*/
g.c3shoot(phi,lambd,Nsampl,usampl,useRad,taylOrd) ;

}
} /* Geoid */

```

-
- [1] Abramowitz, M., and I. A. Stegun (eds.), 1972, *Handbook of Mathematical Functions* (Dover Publications, New York), 9th edition, ISBN 0-486-61272-4.
 - [2] Bouasse, H., 1919, *Geographie Mathématique* (Librairie Delagrave, Paris).
 - [3] Bowring, B. R., 1983, *Bull. Geod.* **57**(1-4), 109.
 - [4] Byrd, P. F., and M. D. Friedman, 1971, *Handbook of elliptical Integrals for Engineers and Physicists*, volume LXVII of *Die Grundlehren der mathematischen Wissenschaften in Einzeldarstellungen* (Springer, Berlin, Göttingen), 2nd edition.
 - [5] Dorrer, E., 1999, in *Quo vadis geodesia...?*, edited by F. Krumm and V. S. Schwarze (Universität Stuttgart), Technical Reports of Geodesy and GeoInformatics, ISSN 0933-2839, URL http://www.uni-stuttgart.de/gi/research/schriftenreihe/quo_vadis/contents.html.
 - [6] Dufour, H. M., 1958, *Bull. Geod.* **32**(2), 26.
 - [7] Fukushima, T., 2006, *J. Geod.* **79**(12), 689.
 - [8] Gradstein, I., and I. Ryshik, 1981, *Summen-, Produkt- und Integraltafeln* (Harri Deutsch, Thun), 1st edition, ISBN 3-87144-350-6.

- [9] Hradilek, L., 1976, *Bull. Geod.* **50**(4), 301.
- [10] Jones, G. C., 2004, *J. Geod.* **76**(8), 437.
- [11] Kaplan, G. H., 2006, arXiv:astro-ph/0602086 .
- [12] Long, S. A. T., 1975, *Cel. Mech.* **12**(2), 225.
- [13] McCarthy, D. D., and G. Petit, 2003, *IERS Technical Note No 32*, Technical Report, IERS Convention Centre, URL <http://www.iers.org/iers/publications/tn/tn32/>.
- [14] National Imagery and Mapping Agency, 2000, *Department Of Defense World Geodetic System 1984*, Technical Report TR8350.2, NIMA, URL <http://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350.2.html>.
- [15] Ölander, V. R., 1952, *Bull. Geod.* **26**(3), 337.
- [16] Pollard, J., 2002, *J. Geod.* **76**(1), 36.
- [17] Reichardt, H., 1957, *Vorlesungen über Vektor- und Tensorrechnung*, volume 34 of *Hochschulbücher für Mathematik* (Deutscher Verlag der Wissenschaften, Berlin).
- [18] Saito, T., 1970, *Bull. Geod.* **44**(4), 341.
- [19] Smart, W. M., 1949, *Text-book on Spherical Astronomy* (Cambridge University Press, Cambridge), 4th edition.
- [20] Sodano, E. M., 1958, *Bull. Geod.* **32**(2), 13.
- [21] Thomas, P. D., 1965, *J. Geophys. Res.* **70**(14), 3331.
- [22] Tienstra, J. M., 1951, *Bull. Geod.* **25**(1), 7.
- [23] Vermeille, H., 2004, *J. Geod.* **78**(1–2), 94.
- [24] You, R.-J., 1999, in *Quo vadis geodesia...?*, edited by F. Krumm and V. S. Schwarze (Universität Stuttgart), Technical Reports of Geodesy and GeoInformatics, ISSN 0933-2839, URL http://www.uni-stuttgart.de/gi/research/schriftenreihe/quo_vadis/contents.html.
- [25] Zhang, C.-D., H. T. Hsu, X. P. Wu, S. S. Li, Q. B. Wang, H. Z. Chai, and L. Du, 2005, *J. Geod.* **79**(8), 413.